Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 3, 2018

J. Uberti
Google

March 2, 2018

WebRTC Forward Error Correction Requirements
draft-ietf-rtcweb-fec-08

Abstract

This document provides information and requirements for how Forward Error Correction (FEC) should be used by WebRTC implementations.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 3, 2018.

Copyright Notice

Table of Contents

1. Introduction

In situations where packet loss is high, or perfect media quality is essential, Forward Error Correction (FEC) can be used to proactively recover from packet losses. This specification provides guidance on which FEC mechanisms to use, and how to use them, for WebRTC implementations.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Types of FEC

FEC describes the sending of redundant information in an outgoing packet stream so that information can still be recovered even in the face of packet loss. There are multiple ways in which this can be

accomplished; this section enumerates the various mechanisms and describes their tradeoffs.

3.1. Separate FEC Stream

This approach, as described in [RFC5956], Section 4.3, sends FEC packets as an independent SSRC-multiplexed stream, with its own SSRC and payload type. While this approach can protect multiple packets of the primary encoding with a single FEC packet, each FEC packet will have its own IP+UDP+RTP+FEC header, and this overhead can be excessive in some cases, e.g., when protecting each primary packet with a FEC packet.

This approach allows for recovery of entire RTP packets, including the full RTP header.

3.2. Redundant Encoding

This approach, as descibed in [RFC2198], allows for redundant data to be piggybacked on an existing primary encoding, all in a single packet. This redundant data may be an exact copy of a previous packet, or for codecs that support variable-bitrate encodings, possibly a smaller, lower-quality representation. In certain cases, the redundant data could include multiple prior packets.

Since there is only a single set of packet headers, this approach allows for a very efficient representation of primary + redundant data. However, this savings is only realized when the data all fits into a single packet (i.e. the size is less than a MTU). As a result, this approach is generally not useful for video content.

As described in [RFC2198], Section 4, this approach cannot recover certain parts of the RTP header, including the marker bit, CSRC information, and header extensions.

3.3. Codec-Specific In-band FEC

Some audio codecs, notably Opus [RFC6716] and AMR [RFC4867], support their own in-band FEC mechanism, where redundant data is included in the codec payload.

For Opus, packets deemed as important are re-encoded at a lower bitrate and added to the subsequent packet, allowing partial recovery of a lost packet. This scheme is fairly efficient; experiments performed indicate that when Opus FEC is used, the overhead imposed is about 20-30%, depending on the amount of protection needed. Note that this mechanism can only carry redundancy information for the immediately preceding packet; as such the decoder cannot fully

recover multiple consecutive lost packets, which can be a problem on wireless networks. See [RFC6716], Section 2.1.7 for complete details.

For AMR/AMR-WB, packets can contain copies or lower-quality encodings of multiple prior audio frames. This mechanism is similar to the [RFC2198] mechanism described above, but as it adds no additional framing, it can be slightly more efficient. See [RFC4867], Section 3.7.1 for details on this mechanism.

In-band FEC mechanisms cannot recover any of the RTP header.

4. FEC for Audio Content

The following section provides guidance on how to best use FEC for transmitting audio data. As indicated in Section 8 below, FEC should only be activated if network conditions warrant it, or upon explicit application request.

4.1. Recommended Mechanism

When using variable-bitrate codecs without an internal FEC, [RFC2198] redundant encoding with lower-fidelity version(s) of the previous packet(s) is RECOMMENDED. This provides reasonable protection of the payload with only moderate bitrate increase, as the redundant encodings can be significantly smaller than the primary encoding.

When using the Opus codec, use of the built-in Opus FEC mechanism is RECOMMENDED. This provides reasonable protection of the audio stream against individual losses, with minimal overhead. Note that, as indicated above, the built-in Opus FEC only provides single-frame redundancy; if multi-packet protection is needed, the aforementioned [RFC2198] redundancy with reduced-bitrate Opus encodings SHOULD be used instead.

When using the AMR/AMR-WB codecs, use of their built-in FEC mechanism is RECOMMENDED. This provides slightly more efficient protection of the audio stream than [RFC2198].

When using constant-bitrate codecs, e.g. PCMU, use of [RFC2198] redundant encoding MAY be used, but note that this will result in a potentially significant bitrate increase, and that suddenly increasing bitrate to deal with losses from congestion may actually make things worse.

Because of the lower packet rate of audio encodings, usually a single packet per frame, use of a separate FEC stream comes with a higher overhead than other mechanisms, and therefore is NOT RECOMMENDED.

As mentioned above, the recommended mechanisms do not allow recovery of parts of the RTP header that may be important in certain audio applications, e.g., CSRCs and RTP header extensions like those specified in [RFC6464] and [RFC6465]. Implementations SHOULD account for this and attempt to approximate this information, using an approach similar to those described in [RFC2198], Section 4, and [RFC6464], Section 5.

4.2. Negotiating Support

Support for redundant encoding of a given RTP stream SHOULD be indicated by including audio/red [RFC2198] as an additional supported media type for the associated m= section in the SDP offer [RFC3264]. Answerers can reject the use of redundant encoding by not including the audio/red media type in the corresponding m= section in the SDP answer.

Support for codec-specific FEC mechanisms are typically indicated via "a=fmtp" parameters.

For Opus, a receiver MUST indicate that it is prepared to use incoming FEC data with the "useinbandfec=1" parameter, as specified in [RFC7587]. This parameter is declarative and can be negotiated separately for either media direction.

For AMR/AMR-WB, support for redundant encoding, and the maximum supported depth, are controlled by the ’max-red’ parameter, as specified in [RFC4867], Section 8.1. Receivers MUST include this parameter, and set it to an appropriate value, as specified in [TS.26114], Table 6.3.

5. FEC for Video Content

The following section provides guidance on how to best use FEC for transmitting video data. As indicated in Section 8 below, FEC should only be activated if network conditions warrant it, or upon explicit application request.

5.1. Recommended Mechanism

Video frames, due to their size, often require multiple RTP packets. As discussed above, a separate FEC stream can protect multiple packets with a single FEC packet. In addition, the "flexfec" FEC mechanism described in [I-D.ietf-payload-flexible-fec-scheme] is also capable of protecting multiple RTP streams via a single FEC stream, including all the streams that are part of a BUNDLE [I-D.ietf-mmusic-sdp-bundle-negotiation] group. As a result, for

video content, use of a separate FEC stream with the flexfec RTP payload format is RECOMMENDED.

To process the incoming FEC stream, the receiver can demultiplex it by SSRC, and then correlate it with the appropriate primary stream(s) via the CSRC(s) present in the RTP header of flexfec repair packets, or the SSRC field present in the FEC header of flexfec retransmission packets.

5.2. Negotiating Support

Support for a SSRC-multiplexed flexfec stream to protect a given RTP stream SHOULD be indicated by including one of the formats described in [I-D.ietf-payload-flexible-fec-scheme], Section 5.1, as an additional supported media type for the associated m= section in the SDP offer [RFC3264]. As mentioned above, when BUNDLE is used, only a single flexfec repair stream will be created for each BUNDLE group, even if flexfec is negotiated for each primary stream.

Answerers can reject the use of SSRC-multiplexed FEC, by not including the offered FEC formats in the corresponding m= section in the SDP answer.

Use of FEC-only m-lines, and grouping using the SDP group mechanism as described in [RFC5956], Section 4.1 is not currently defined for WebRTC, and SHOULD NOT be offered.

Answerers SHOULD reject any FEC-only m-lines, unless they specifically know how to handle such a thing in a WebRTC context (perhaps defined by a future version of the WebRTC specifications).

6. FEC for Application Content

WebRTC also supports the ability to send generic application data, and provides transport-level retransmission mechanisms to support full and partial (e.g. timed) reliability. See [I-D.ietf-rtcweb-data-channel] for details.

Because the application can control exactly what data to send, it has the ability to monitor packet statistics and perform its own application-level FEC, if necessary.

As a result, this document makes no recommendations regarding FEC for the underlying data transport.

7. Implementation Requirements

To support the functionality recommended above, implementations MUST be able to receive and make use of the relevant FEC formats for their supported audio codecs, and MUST indicate this support, as described in Section 4. Use of these formats when sending, as mentioned above, is RECOMMENDED.

The general FEC mechanism described in [I-D.ietf-payload-flexible-fec-scheme] SHOULD also be supported, as mentioned in Section 5.

Implementations MAY support additional FEC mechanisms if desired, e.g., [RFC5109].

8. Adaptive Use of FEC

Because use of FEC always causes redundant data to be transmitted, and the total amount of data must remain within any bandwidth limits indicated by congestion control and the receiver, this will lead to less bandwidth available for the primary encoding, even when the redundant data is not being used. This is in contrast to methods like RTX [RFC4588] or flexfec [I-D.ietf-payload-flexible-fec-scheme] retransmissions, which only transmit redundant data when necessary, at the cost of an extra roundtrip.

Given this, WebRTC implementations SHOULD consider using RTX or flexfec retransmissions instead of FEC when RTT is low, and SHOULD only transmit the amount of FEC needed to protect against the observed packet loss (which can be determined, e.g., by monitoring transmit packet loss data from RTCP Receiver Reports [RFC3550]), unless the application indicates it is willing to pay a quality penalty to proactively avoid losses.

Note that when probing bandwidth, i.e., speculatively sending extra data to determine if additional link capacity exists, FEC SHOULD be used in all cases. Given that extra data is going to be sent regardless, it makes sense to have that data protect the primary payload; in addition, FEC can be applied in a way that increases bandwidth only modestly, which is necessary when probing.

When using FEC with layered codecs, e.g., [RFC6386], where only base layer frames are critical to the decoding of future frames, implementations SHOULD only apply FEC to these base layer frames.

9. Security Considerations

This document makes recommendations regarding the use of FEC. Generally, it should be noted that although applying redundancy is often useful in protecting a stream against packet loss, if the loss is caused by network congestion, the additional bandwidth used by the redundant data may actually make the situation worse, and can lead to significant degradation of the network.

As described in [RFC3711], Section 10, the default processing when using FEC with SRTP is to perform FEC followed by SRTP at the sender, and SRTP followed by FEC at the receiver. This ordering is used for all the SRTP Protection Profiles used in DTLS-SRTP [RFC5763], as described in [RFC5764], Section 4.1.2.

Additional security considerations for each individual FEC mechanism are enumerated in their respective documents.

10. IANA Considerations

This document requires no actions from IANA.

11. Acknowledgements

Several people provided significant input into this document, including Bernard Aboba, Jonathan Lennox, Giri Mandyam, Varun Singh, Tim Terriberry, Magnus Westerlund, and Mo Zanaty.

12. References

12.1. Normative References

[I-D.ietf-payload-flexible-fec-scheme]
Singh, V., Begen, A., Zanaty, M., and G. Mandyam, "RTP Payload Format for Flexible Forward Error Correction (FEC)", draft-ietf-payload-flexible-fec-scheme-05 (work in progress), July 2017.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2198] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J., Vega-Garcia, A., and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data", RFC 2198, DOI 10.17487/RFC2198, September 1997, <https://www.rfc-editor.org/info/rfc2198>.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <https://www.rfc-editor.org/info/rfc3264>.

[RFC4867] Sjoberg, J., Westerlund, M., Lakaniemi, A., and Q. Xie, "RTP Payload Format and File Storage Format for the Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (AMR-WB) Audio Codecs", RFC 4867, DOI 10.17487/RFC4867, April 2007, <https://www.rfc-editor.org/info/rfc4867>.

[RFC5956] Begen, A., "Forward Error Correction Grouping Semantics in the Session Description Protocol", RFC 5956, DOI 10.17487/RFC5956, September 2010, <https://www.rfc-editor.org/info/rfc5956>.

[RFC7587] Spittka, J., Vos, K., and JM. Valin, "RTP Payload Format for the Opus Speech and Audio Codec", RFC 7587, DOI 10.17487/RFC7587, June 2015, <https://www.rfc-editor.org/info/rfc7587>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[TS.26114] 3GPP, "IP Multimedia Subsystem (IMS); Multimedia telephony; Media handling and interaction", 3GPP TS 26.114 15.0.0, September 2017.

12.2. Informative References

[I-D.ietf-mmusic-sdp-bundle-negotiation] Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-negotiation-48 (work in progress), January 2018.

[I-D.ietf-rtcweb-data-channel] Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", draft-ietf-rtcweb-data-channel-13 (work in progress), January 2015.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <https://www.rfc-editor.org/info/rfc3711>.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, DOI 10.17487/RFC4588, July 2006, <https://www.rfc-editor.org/info/rfc4588>.

[RFC5109] Li, A., Ed., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, DOI 10.17487/RFC5109, December 2007, <https://www.rfc-editor.org/info/rfc5109>.

[RFC5763] Fischl, J., Tschofenig, H., and E. Rescorla, "Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS)", RFC 5763, DOI 10.17487/RFC5763, May 2010, <https://www.rfc-editor.org/info/rfc5763>.

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, DOI 10.17487/RFC5764, May 2010, <https://www.rfc-editor.org/info/rfc5764>.

[RFC6386] Bankoski, J., Koleszar, J., Quillio, L., Salonen, J., Wilkins, P., and Y. Xu, "VP8 Data Format and Decoding Guide", RFC 6386, DOI 10.17487/RFC6386, November 2011, <https://www.rfc-editor.org/info/rfc6386>.

[RFC6464] Lennox, J., Ed., Ivov, E., and E. Marocco, "A Real-time Transport Protocol (RTP) Header Extension for Client-to-Mixer Audio Level Indication", RFC 6464, DOI 10.17487/RFC6464, December 2011, <https://www.rfc-editor.org/info/rfc6464>.

[RFC6465] Ivov, E., Ed., Marocco, E., Ed., and J. Lennox, "A Real-time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication", RFC 6465, DOI 10.17487/RFC6465, December 2011, <https://www.rfc-editor.org/info/rfc6465>.

[RFC6716] Valin, JM., Vos, K., and T. Terriberry, "Definition of the Opus Audio Codec", RFC 6716, DOI 10.17487/RFC6716, September 2012, <https://www.rfc-editor.org/info/rfc6716>.

Appendix A. Change log

Changes in draft -08:

- Switch to RFC 8174 boilerplate.

Changes in draft -07:

- Clarify how bandwidth management interacts with FEC.
- Make 3GPP reference normative.

Changes in draft -06:

- Discuss how multiple streams can be protected by a single FlexFEC stream.
- Discuss FEC for bandwidth probing.
- Add note about recovery of RTP headers and header extensions.
- Add note about FEC/SRTP ordering.
- Clarify flexfec demux text, and mention retransmits.
- Clarify text regarding offers/answers.
- Make RFC2198 support SHOULD strength.
- Clean up references.

Changes in draft -05:

- No changes.

Changes in draft -04:

- Discussion of layered codecs.
- Discussion of RTX.
- Clarified implementation requirements.
- FlexFEC MUST -> SHOULD.
- Clarified AMR max-red handling.
- Updated references.

Changes in draft -03:

- Added overhead stats for Opus.
- Expanded discussion of multi-packet FEC for Opus.
- Added discussion of AMR/AMR-WB.
- Removed discussion of ssrc-group.
- Referenced the data channel doc.
- Referenced the RTP/RTCP RFC.
- Several small edits based on feedback from Magnus.

Changes in draft -02:

- Expanded discussion of FEC-only m-lines, and how they should be handled in offers and answers.

Changes in draft -01:

- Tweaked abstract/intro text that was ambiguously normative.
- Removed text on FEC for Opus in CELT mode.
- Changed RFC 2198 recommendation for PCMU to be MAY instead of NOT RECOMMENDED, based on list feedback.
- Explicitly called out application data as something not addressed in this document.
- Updated flexible-fec reference.

Changes in draft -00:

- Initial version, from sidebar conversation at IETF 90.

Author's Address

Justin Uberti
Google
747 6th St S
Kirkland, WA 98033
USA

Email: justin@uberti.name

RTCWeb Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 24, 2016

H. Alvestrand
Google
U. Rauschenbach
Nokia Networks

January 21, 2016

WebRTC Gateways
draft-ietf-rtcweb-gateways-02

Abstract

This document describes interoperability considerations for a class of WebRTC-compatible endpoints called "WebRTC gateways", which interconnect between WebRTC endpoints and devices that are not WebRTC endpoints.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 24, 2016.

Copyright Notice

Table of Contents

1. Introduction

The WebRTC model described in [I-D.ietf-rtcweb-overview] is focused on direct browser to browser communication as its primary use case. Nevertheless, it is clearly interesting to have WebRTC endpoints connect to other types of devices, including but not limited to SIP phones, legacy phones, CLUE-based teleconferencing systems, XMPP-based conferencing systems, and entirely proprietary devices or systems.

WebRTC gateways are middle boxes which enable the exchange of media streams between WebRTC endpoints on one side, and the other types of devices mentioned above on the other side. To a WebRTC endpoint, the gateway appears as a WebRTC-compatible endpoint.

This document describes the requirements that need to be placed on such gateways, both the requirements on WebRTC endpoints that can be relaxed and the additional requirements that need to be applied.

A WebRTC gateway appears as a WebRTC-compatible endpoint, and will thus not be conformant with all requirements for a WebRTC endpoint (it does not do everything a WebRTC endpoint does), but is able to interoperate with WebRTC endpoints.

NOTE IN DRAFT: There is still not a WG consensus called on whether this document is Informational or standards-track. If it becomes informational, the use of RFC 2119 language is used to call attention to features where non-conformance will render a gateway unable to interoperate with WebRTC-based endpoints.

1.1. Implications of the gateway environment

A gateway will be limited in the functionality it can offer by the system or class of devices it is gatewaying to. For instance, a gateway into the telephone system will not be able to relay data or video, no matter how much it is required. Therefore, a number of functions that are mandatory to support in WebRTC endpoints are not mandatory on gateways; the requirement on the gateway is that it is able to negotiate those features away correctly.

1.2. Signalling model

The WebRTC model is that signalling is outside the scope of the specification. This document does not change that.

Nevertheless, any practical gateway needs to deal with signalling. For that, this document assumes that the overall system consists of an application running in the WebRTC browser, possibly one or more signalling relays that mediate signalling and thereby enable communication between the application and the gateway, and the actual gateway that is responsible for handling the media flows.

The application, the signalling relays (if any) and the gateway together need to be able to:

- adhere to the offer/answer semantics
- deal with the description of configuration coming from the browser; this is specified in SDP format in the WebRTC browser API
- generate the information that is needed by the browser to set up the session, and express that information in the form of SDP.

The shorthand notation "The gateway MUST/SHOULD/MAY support <SDP function xxx>" used below means that an application running in the Web browser, the signalling relays, and the gateway together MUST/SHOULD/MAY support this functionality; it is not a requirement that this happens at the media gateway itself.

2. WebRTC non-browser requirements that can be relaxed

WebRTC gateways are intended to communicate with WebRTC endpoints[I-D.ietf-rtcweb-overview]. Some features that typical WebRTC endpoints are required to support may be meaningless or unneccesary for WebRTC gateways; some such things are noted in this section. This lack of conformance means that a gateway is considered a WebRTC-compatible endpoint, not a WebRTC endpoint (unless a particular gateway claims to be a WebRTC endpoint, which it is of course allowed to do).

A WebRTC gateway which is expected to be deployed where it can be reached with a static IP address (as seen from the client) does not need to support full ICE; it therefore MAY implement ICE-Lite only.

ICE-Lite implementations do not send consent checks, so a gateway MAY choose not to send consent checks too, but MUST respond to consent checks it receives.

A gateway with a static IP address is expected to not need to hide its location, so it does not need to support functionality for operating only via a TURN server; instead it MAY choose to produce Host ICE candidates only.

If a gateway serves as a media relay into another RTP domain, it MAY choose to support only features available in that network. This means that it MAY choose to not support Bundle and any of the RTP/RTCP extensions related to it, RTCP-Mux, or Trickle Ice. However, the gateway MUST support DTLS-SRTP, since this is required for interworking with WebRTC endpoints.

Note that non-support of BUNDLE means that "bundle-only" tracks are not supported. This means that applications using an RTCBundlePolicy other than "max-compat" ([I-D.ietf-rtcweb-jsep] section 4.1.1) can only use one track of each media type.

If a gateway serves as a media relay into a network or to devices not implementing the WebRTC Datachannel, it MAY choose to not support the Datachannel.

3. Additional WebRTC gateway requirements

(nothing yet)

4. Considerations for SDP-using networks

Some networks that are gatewayed into, such as SIP networks, will also use SDP to represent the media configurations. Gateways will, however, need to inspect and probably modify the SDP passed between the SDP-using network and the WebRTC endpoints to achieve maximum interoperability.

Considerations include:

- If a correspondent does not offer the features WebRTC depends on, connections will not complete. The support for dtls-srtp, shown by the "fingerprint" attribute, is the most obvious example. The gateway is probably better off either ending such calls early or acting as a full B2BUA (as defined in [RFC3261]) with media gatewaying.

- If a correspondent makes an offer using features that are not required by JSEP, these may not be understood by the WebRTC implementation. The gateway may choose to strip out some such features.

- Certain ancient practices (such as using port 0 to place a media section on hold with the intent of resuming it later) are not conformant with the SDP offer/answer spec ([RFC3264] section 8.2). Since WebRTC implementations are expected to be SDP offer/answer conformant, such practices may need to be stripped out by the gateway

[NOTE IN DRAFT: This section may need expanding.]

5. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

6. Security Considerations

A WebRTC gateway may operate in two security modes: Security-context termination and security-context relaying.

Relaying is only possible where signed and encrypted content can be passed through unchanged, and where keys can be exchanged directly between the endpoints.

When the gateway terminates the security context, it means that the WebRTC user has to place trust in the gateway to perform all verification of identity and protection of content in the realm on the other side of the gateway; there is no way the end-user can detect a man-in-the-middle attack, an identity spoofing attack or a recording done at the gateway. For many scenarios, this is not going to be seen as a problem, but needs to be considered when one decides to use a gatewayed service.

7. Acknowledgements

Several comments from Christer Holmberg and Andrew Hutton were included.

8. Change history

Changes from draft-alvestrand-rtcweb-gateways-00

- Aligned terminology with draft-rtcweb-overview-12
- Rewrote text on signaling to improve clarity
- Editorial nits

Changes from draft-alvestrand-rtcweb-gateways-01

- Aligned terminology with draft-rtcweb-overview-13 ("non-browser")
- Nits

Changes from draft-alvestrand-rtcweb-gateways-02

- Re-submitted as WG draft
- Addressed a comment from Andrew Hutton that deployment in open internet is an option, not a fact.

Changes from draft-ietf-rtcweb-gateways-00

- Added note about impllications of non-support of BUNDLE
- Added "Considerations for SDP-using networks" section

Changes from draft-ietf-rtcweb-gateways-01: None, this is a keepalive update.

9. References

9.1. Normative References

[I-D.ietf-rtcweb-jsep]
Uberti, J., Jennings, C., and E. Rescorla, "Javascript Session Establishment Protocol", draft-ietf-rtcweb-jsep-09 (work in progress), March 2015.

[I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-13 (work in progress), November 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Informative References

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

Authors' Addresses

Harald Alvestrand
Google

Email: harald@alvestrand.no

Uwe Rauschenbach
Nokia Networks

Email: uwe.rauschenbach@nokia.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 14, 2015

B. Schwartz
J. Uberti
Google

May 13, 2015

Recursively Encapsulated TURN (RETURN) for Connectivity and Privacy in WebRTC
draft-ietf-rtcweb-return-00

Abstract

In the context of WebRTC, the concept of a local TURN proxy has been suggested, but not reviewed in detail. WebRTC applications are already using TURN to enhance connectivity and privacy. This document explains how local TURN proxies and WebRTC applications can work together.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 14, 2015.

Copyright Notice

Table of Contents

1. Introduction

TURN [RFC5766] is a protocol for communication between a client and a TURN server, in order to route UDP traffic to and from one or more peers. As noted in [RFC5766], the TURN relay server "typically sits in the public Internet". In a WebRTC context, if a TURN server is to be used, it is typically provided by the application, either to provide connectivity between users whose NATs would otherwise prevent

it, or to obscure the identity of the participants by concealing their IP addresses from one another.

In many enterprises, direct UDP transmissions are not permitted between clients on the internal networks and external IP addresses, so media must flow over TCP. To enable WebRTC services in such a situation, clients must use TURN-TCP, or TURN-TLS. These configurations are not ideal: they send all traffic over TCP, which leads to higher latency than would otherwise be necessary, and they force the application provider to operate a TURN server because WebRTC endpoints behind NAT cannot typically act as TCP servers. These configurations may result in especially bad behaviors when operating through TCP or HTTP proxies that were not designed to carry real-time media streams.

To avoid forcing WebRTC media streams through a TCP stage, enterprise network operators may operate a TURN server for their network, which can be discovered by clients using TURN Auto-Discovery [I-D.ietf-tram-turn-server-discovery], or through a proprietary mechanism. This TURN server may be placed inside the network, with a firewall configuration allowing it to communicate with the public internet, or it may be operated by a third party outside the network, with a firewall configuration that allows hosts inside the network to communicate with it. Use of the specified TURN server may be the only way for clients on the network to achieve a high quality WebRTC experience. This scenario is required to be supported by the WebRTC requirements document [I-D.ietf-rtcweb-use-cases-and-requirements] Section 3.3.5.1.

When the application intends to use a TURN server for identity cloaking, and the enterprise network administrator intends to use a TURN server for connectivity, there is a conflict. In current WebRTC implementations, TURN can only be used on a single-hop basis in each candidate, but using only the enterprise’s TURN server reveals information about the user (e.g. organizational affiliation), and using only the application’s TURN server may be blocked by the network administrator, or may require using TURN-TCP or TURN-TLS, resulting in a significant sacrifice in latency.

To resolve this conflict, we introduce Recursively Encapsulated TURN, a procedure that allows a WebRTC endpoint to route traffic through multiple TURN servers, and get improved connectivity and privacy in return.

2. Visual Overview of RETURN

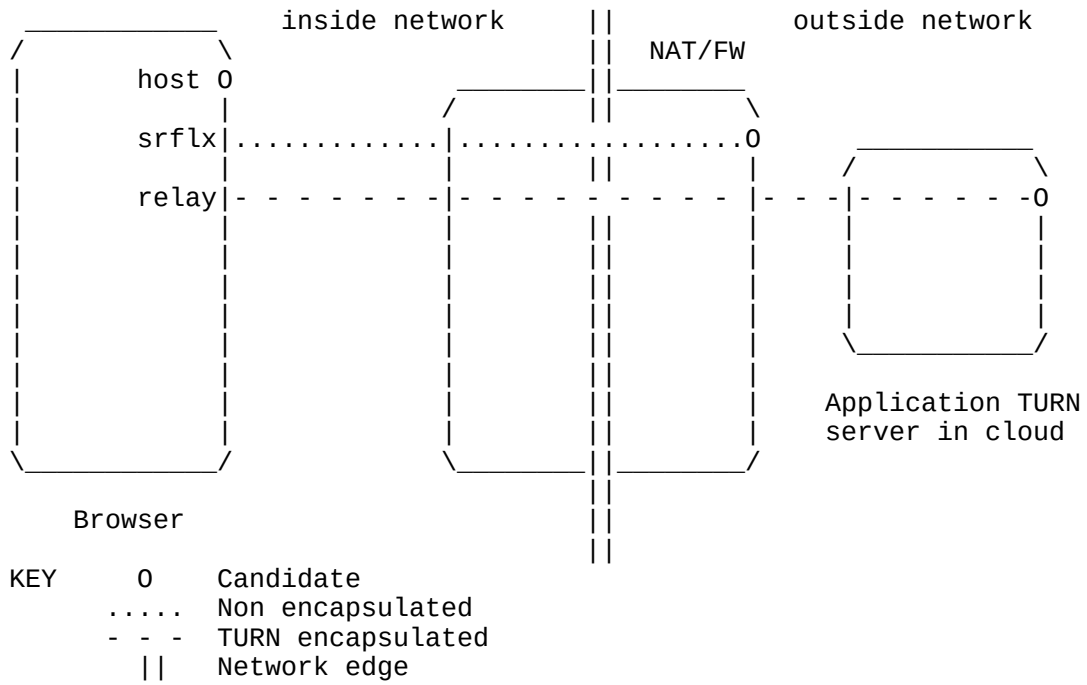


Figure 1: Basic WebRTC ICE Candidates with TURN Server

Figure 1 shows a browser located inside a home or enterprise network which connects to the Internet through a Network Address Translator and Firewall (NAT/FW). A TURN server in the Internet cloud is also shown, which is provided by the WebRTC application via the JavaScript IceServers object.

A WebRTC application can use a TURN server to provide NAT traversal, but also to provide privacy, routing optimizations, logging, or possibly other functionality. The application can accomplish this by forcing all traffic to flow through the TURN server using the JavaScript RTCIceTransportPolicy object [I-D.ietf-rtcweb-jsep]. Since this TURN server is injected by the application, we will refer to it as an Application TURN server.

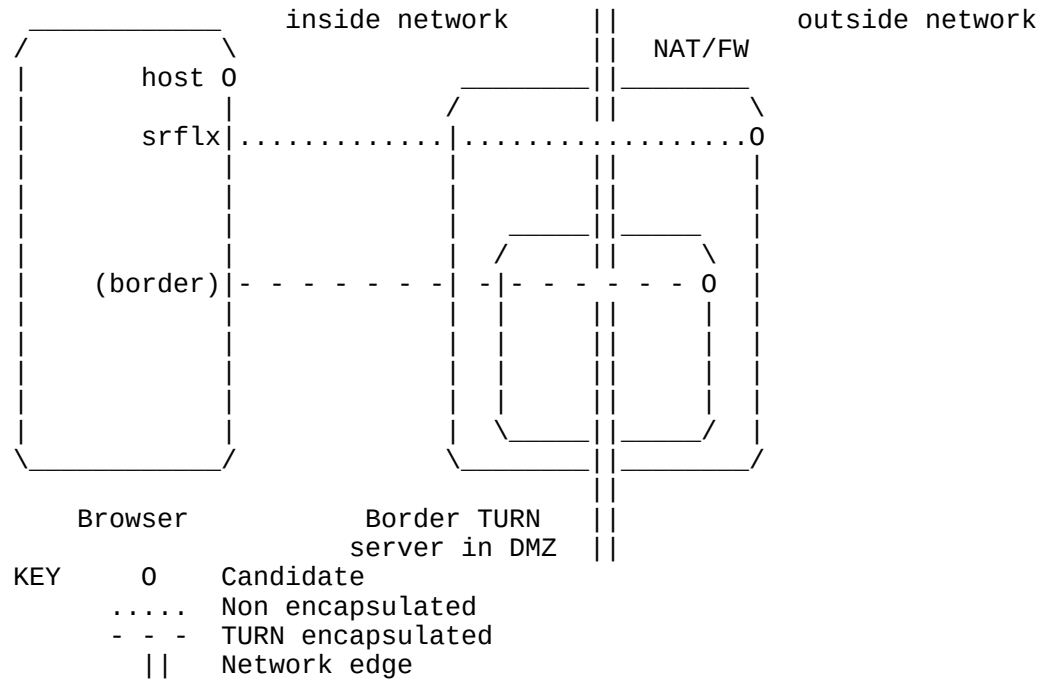


Figure 2: WebRTC ICE Candidates with DMZ TURN Server

Figure 2 shows a TURN server co-resident with the NAT/FW, i.e. in the DMZ of the FW. This TURN server might be used by an enterprise, ISP, or home network to enable WebRTC media flows that would otherwise be blocked by the firewall, or to improve quality of service on flows that pass through this TURN server. This TURN server is not part of a particular application, and is managed as part of the border control system, so we call it a Border TURN Server.

Figure 2 shows the port allocated on this TURN server as "(border)", not any particular candidate type, to distinguish it from the other ports, which have been represented as ICE candidates in accordance with the WebRTC specifications. This case is different, because unlike an Application TURN server, there is not yet any specification for how WebRTC should interact with a Border TURN server. Under what conditions should WebRTC allocate a port on a Border TURN server? How should WebRTC represent that port as an ICE candidate? This draft serves to answer these two questions.

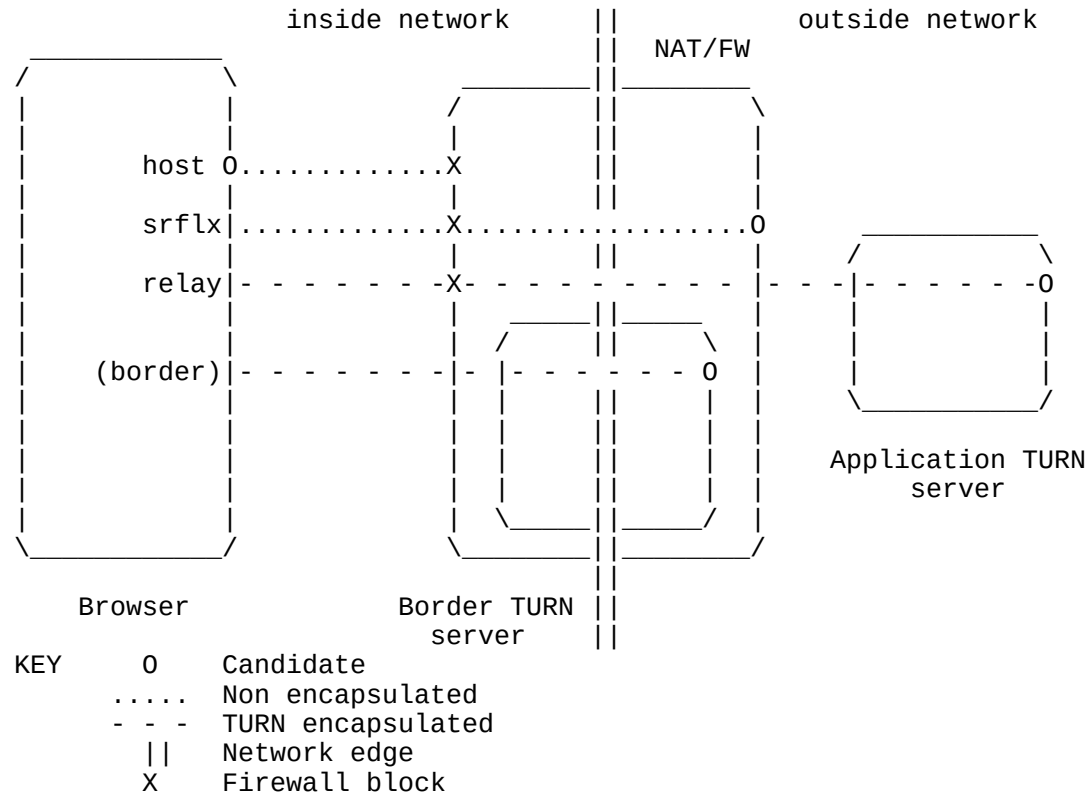


Figure 3: WebRTC ICE Candidates with Application and Border TURN Servers

In Figure 3, there is both an Application TURN server and a Border TURN server. The Firewall is blocking UDP traffic except for UDP traffic to/from the Border TURN server, so only the "(border)" port allocation will work. However, there is no specified way for WebRTC to use this port as a candidate. Moreover, this port on its own would not be sufficient to satisfy the user's needs. Both TURN servers provide important functionality, so we need a way for WebRTC to select a candidate that uses both TURN servers.

The solution proposed in this draft is for the browser to implement RETURN, which provides a candidate that traverses both TURN servers, as shown in Figure 4.

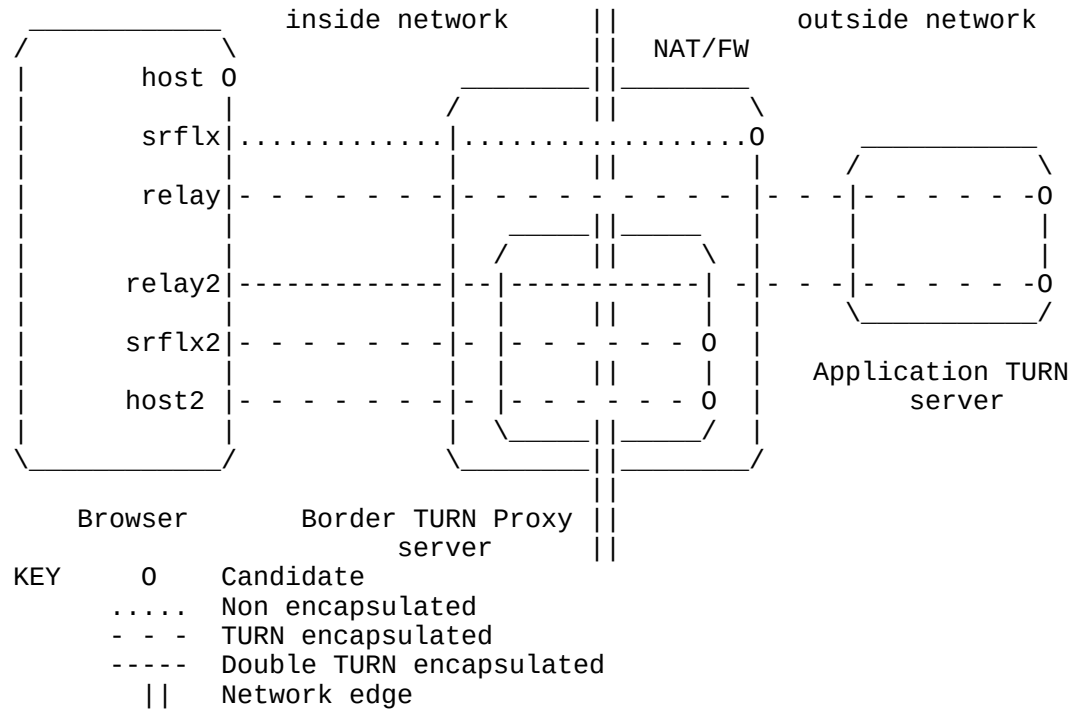


Figure 4: WebRTC ICE Candidates with Application TURN and Border TURN Proxy Servers

The Browser in Figure 4 implements RETURN, so it allocates a port on the Border TURN server, now referred to as a Border TURN Proxy by analogy to an HTTP CONNECT or SOCKS Proxy (see Figure 5), and then runs STUN and TURN over this allocation, resulting in three candidates: relay2, srflx2, and host2. The relay2 candidate causes traffic to flow through both TURN servers by encapsulating TURN within TURN - hence the name Recursively Encapsulated TURN (RETURN).

The host2 and srflx2 candidates are probably identical, so one will be dropped by ICE. If the NAT/FW blocks UDP and the application uses only relay candidates, then the relay2 candidate will be selected. Otherwise, the other candidates will be used, in accordance with the usual ICE procedure.

Only the browser needs to implement the RETURN behavior - both the Border TURN Proxy and Application TURN servers' TURN protocol usage is unchanged.

Note that this arrangement preserves the end-to-end security and privacy features of WebRTC media flows. The ability to steer the

media flows through multiple TURN servers while still allowing end-to-end encryption and authentication is a key benefit of RETURN.

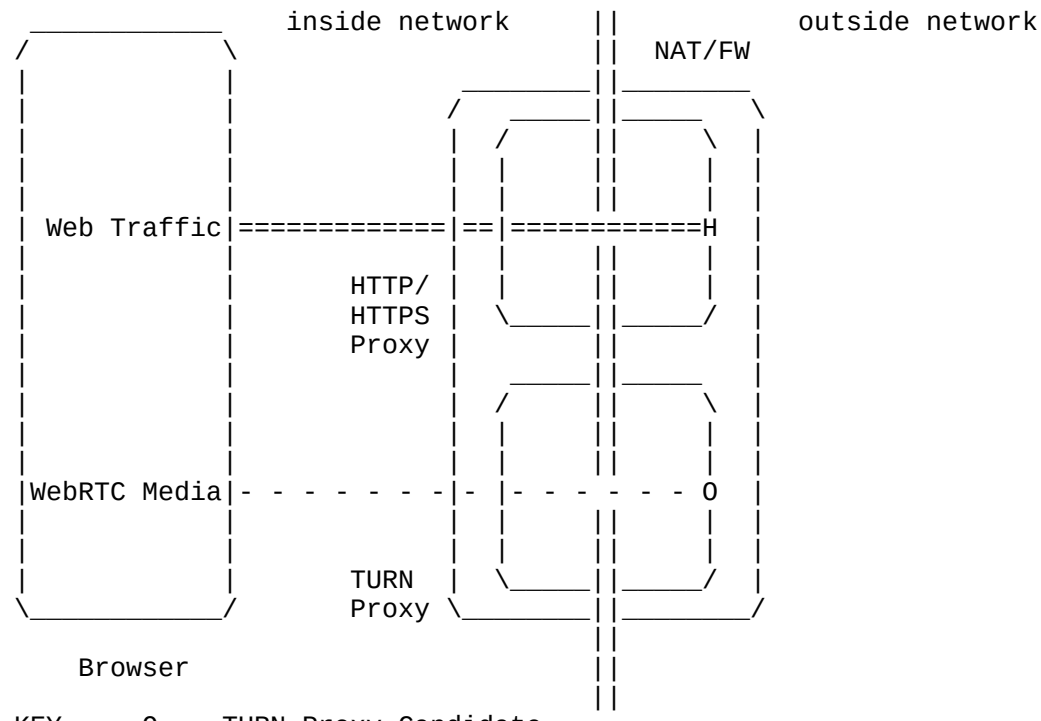


Figure 5: Similarity between HTTP/HTTPS Proxy and TURN Proxy

3. Goals

These goals are requirements on this document (not on implementations of the specification).

3.1. Connectivity

As noted in [I-D.ietf-rtcweb-use-cases-and-requirements] Section 3.3.5.1 and requirement F20, a WebRTC browser endpoint MUST be able to direct UDP connections through a designated TURN server configured by enterprise policy (a "proxy").

It MUST be possible to configure a WebRTC endpoint that supports proxies to achieve connectivity no worse than if the endpoint were operating at the proxy's address.

For efficiency, network administrators SHOULD be able to prevent browsers from attempting to send traffic through routes that are already known to be blocked.

3.2. Independent Path Control

Both network administrators and application developers may wish to direct all their UDP flows through a particular TURN server. There are many goals that might motivate such a choice, including

- improving quality of service by tunneling packets through a network that is faster than the public internet,
- monitoring the usage of UDP services,
- troubleshooting and debugging problematic services,
- logging connection metadata for legal or auditing reasons,
- recording the entire contents of all connections, or
- providing partial IP address anonymization (as described in [I-D.ietf-rtcweb-security] Section 4.2.4).

4. Concepts

To achieve our goals, we introduce the following new concepts:

4.1. Proxy

In this document a "proxy" is any TURN server that was provided by any mechanism other than through the standard WebRTC-application ICE candidate provisioning API [I-D.ietf-rtcweb-jsep]. We call it a "proxy" by analogy with SOCKS proxies and similar network services, because it performs a similar function and can be configured in a similar fashion.

If a proxy is to be used, it will be the destination of traffic generated by the client. (There is no analogue to the transparent/ intercepting HTTP proxy configuration, which modifies traffic at the network layer.) Mechanisms to configure a proxy include Auto-Discovery [I-D.ietf-tram-turn-server-discovery] and local policy ([I-D.ietf-rtcweb-jsep], "ICE candidate policy").

In an application context, a proxy may be "active" (producing candidates) or "inactive" (not in use, having no effect on the context).

4.2. Virtual interface

A typical WebRTC browser endpoint may have multiple network interfaces available, such as wired ethernet, wireless ethernet, and WAN. In this document, a "virtual interface" is a procedure for generating ICE candidates that are not simply generated by a particular physical interface. A virtual interface can produce "host", "server-reflexive", and "relay" candidates, but may be restricted to only some type of candidate (e.g. UDP-only).

4.3. Proxy configuration leakiness

"Leakiness" is an attribute of a proxy configuration. This document defines two values for the "leakiness" of a proxy configuration: "leaky" and "sealed". Proxy configuration, including leakiness, may be set by local policy ([I-D.ietf-rtcweb-jsep], "ICE candidate policy") or other mechanisms.

A leaky configuration adds a proxy and also allows the browser to use routes that transit directly via the endpoint’s physical interfaces (not through the proxy). In a leaky configuration, setting a proxy augments the available set of ICE candidates. Multiple leaky-configuration proxies may therefore be active simultaneously.

A sealed proxy configuration requires the browser to route all WebRTC traffic through the proxy, eliminating all ICE candidates that do not go through the proxy. Only one sealed proxy may be active at a time.

Leaky proxy configurations allow more efficient routes to be selected. For example, two peers on the same LAN can connect directly (peer to peer) if a leaky proxy is enabled, but must "hairpin" through the TURN proxy if the configuration is sealed. However, sealed proxy configurations can be faster to connect, especially if many of the peer-to-peer routes that ICE will try first are blocked by the network’s firewall policies.

4.4. Sealed proxy rank

In some configurations, an endpoint may be subject to multiple sealed proxy settings at the same time. In that case, one of those settings will have highest rank, and it will be the active proxy. In a given application context (e.g. a webpage), there is at most one active sealed proxy. This document does not specify a representation for rank.

5. Requirements

5.1. ICE candidates produced in the presence of a proxy

When a proxy is configured, by Auto-Discovery or a proprietary means, the browser MUST NOT report a "relay" candidate representing the proxy. Instead, the browser MUST connect to the proxy and then, if the connection is successful, treat the TURN tunnel as a UDP-only virtual interface.

For a virtual interface representing a TURN proxy, this means that the browser MUST report the public-facing IP address and port acquired through TURN as a "host" candidate, the browser MUST perform STUN through the TURN proxy (if STUN is configured), and it MUST perform TURN by recursive encapsulation through the TURN proxy, resulting in TURN candidates whose "raddr" and "rport" attributes match the acquired public-facing IP address and port on the proxy.

Because the virtual interface has some additional overhead due to indirection, it SHOULD have lower priority than the physical interfaces if physical interfaces are also active. Specifically, even host candidates generated by a virtual interface SHOULD have priority 0 when physical interfaces are active (similar to [RFC5245] Section 4.1.2.2, "the local preference for host candidates from a VPN interface SHOULD have a priority of 0").

5.2. Leaky proxy configuration

If the active proxy for an application is leaky, the browser should undertake the standard ICE candidate discovery mechanism [RFC5245] on the available physical and virtual interfaces.

5.3. Sealed proxy configuration

If the active proxy for an application is sealed, the browser MUST NOT gather or produce any candidates on physical interfaces. The WebRTC implementation MUST direct its traffic from those interfaces only to the proxy, and perform ICE candidate discovery only on the single virtual interface representing the active proxy.

5.4. Proxy rank

Any browser mechanism for specifying a proxy SHOULD allow the caller to indicate a higher rank than the proxy provided by Auto-Discovery [I-D.ietf-tram-turn-server-discovery].

5.5. Multiple physical interfaces

Some operating systems allow the browser to use multiple interfaces to contact a single remote IP address. To avoid producing an excessive number of candidates, WebRTC endpoints MUST NOT use multiple physical interfaces to connect to a single proxy simultaneously. (If this were violated, it could produce a number of virtual interfaces equal to the product of the number of physical interfaces and the number of active proxies.)

For strategies to choose the best interface for communication with a proxy, see [I-D.reddy-mmusic-ice-best-interface-pcp]. Similar considerations apply when connecting to an application-specified TURN server in the presence of physical and virtual interfaces.

5.6. IPv4 and IPv6

A proxy MAY have both an IPv4 and an IPv6 address (e.g. if the proxy is specified by DNS and has both A and AAAA records). The client MAY try both of these addresses, but MUST select one, preferring IPv6, before allocating any remote addresses. This corresponds to the the Happy Eyeballs [RFC6555] procedure for dual-stack clients.

A proxy MAY provide both IPv4 and IPv6 remote addresses to clients [RFC6156]. A client SHOULD request both address families. If both requests are granted, the client SHOULD treat the two addresses as host candidates on a dual-stack virtual interface.

5.7. Unspecified leakiness

If a proxy configuration mechanism does not specify leakiness, browsers SHOULD treat the proxy as leaky. This is similar to current WebRTC implementations' behavior in the presence of SOCKS and HTTP proxies: the candidate allocation code continues to generate UDP candidates that do not transit through the proxy.

5.8. Interaction with SOCKS5-UDP

The SOCKS5 proxy standard [RFC1928] permits compliant SOCKS proxies to support UDP traffic. However, most implementations of SOCKS5 today do not support UDP. Accordingly, WebRTC browsers MUST by default (i.e. unless deliberately configured otherwise) treat SOCKS5 proxies as leaky and having lower rank than any configured TURN proxies.

5.9. Encapsulation overhead, fragmentation, and Path MTU

Encapsulating a link in TURN adds overhead on the path between the client and the TURN server, because each packet must be wrapped in a TURN message. This overhead is sometimes doubled in RETURN proxying. To avoid excessive overhead, client implementations SHOULD use ChannelBind and ChannelData messages to connect and send data through proxies and application TURN servers when possible. Clients MAY buffer messages to be sent until the ChannelBind command completes (requiring one round trip to the proxy), or they MAY use CreatePermission and Send messages for the first few packets to reduce startup latency at the cost of higher overhead.

Adding overhead to packets on a link decreases the effective Maximum Transmissible Unit on that link. Accordingly, clients that support proxying MUST NOT rely on the effective MTU complying with the Internet Protocol's minimum MTU requirement.

ChannelData messages have constant overheard, enabling consistent effective PMTU, but Send messages do not necessarily have constant overhead. TURN messages may be fragmented and reassembled if they are not marked with the Don't Fragment (DF) IP bit or the DONT-FRAGMENT TURN attribute. Client implementors should keep this in mind, especially if they choose to implement PMTU discovery through the proxy.

5.10. Interaction with alternate TURN server fallback

As per [RFC5766], a TURN server MAY respond to an Allocate request with an error code of 300 and an ALTERNATE-SERVER indication. When connecting to proxies or application TURN servers, clients SHOULD attempt to connect to the specified alternate server in accordance with [RFC5766]. The client MUST route a connection to the alternate server through the proxy if and only if the original connection attempt was routed through the proxy.

5.11. Reusing the same TURN server

It is possible that the same TURN server may appear more than once in the network path. For example, if both endpoints configure the same sealed proxy, then each peer will only provide candidates on this proxy. This is not a problem, and will work as expected.

It is also possible that the same TURN server could be used by both the enterprise and the application. It might appear attractive to connect to this server only once, rathering connecting to it through itself, in order to avoid imposing unnecessary server load. However,

a RETURN client MUST connect to the server twice, even when this appears redundant, to ensure correct session attribution.

For example, consider a TURN service operator that issues different authentication credentials to different customers, and then allows each customer to observe the source and destination IP addresses used with their credentials. Suppose the application and enterprise both have accounts on this service: the application uses it to prevent the enterprise from learning its peers' IP addresses, and the enterprise uses it to prevent the application from learning its employees' IP addresses. If the client only connects to the service once, then either the enterprise or the application will learn IP address information (via the TURN provider's metadata reporting) that was meant to be kept secret.

As a result of this requirement, it is possible for the same TURN server to appear up to four times in a RETURN network path: once as each peer's application's TURN server, and once as each peer's sealed proxy.

6. Examples

6.1. Firewalled enterprise network with a basic application

In this example, an enterprise network is configured with a firewall that blocks all UDP traffic, and a TURN server is advertised for Auto-Discovery in accordance with [I-D.ietf-tram-turn-server-discovery]. The proxy leakiness of the TURN server is unspecified, so the browser treats it as leaky.

The application specifies a STUN and TURN server on the public net. In accordance with the ICE candidate gathering algorithm RFC 5245 [RFC5245], it receives a set of candidates like:

1. A host candidate acquired from one interface.
    * e.g. candidate:1610808681 1 udp 2122194687 [internal ip addr for interface 0] 63555 typ host generation 0
2. A host candidate acquired from a different interface.
    * e.g. candidate:1610808681 1 udp 2122194687 [internal ip addr for interface 1] 54253 typ host generation 0
3. The proxy, as a host candidate.
    * e.g. candidate:3458234523 1 udp 24584191 [public ip addr for the proxy] 54606 typ host generation 0

4. The virtual interface also generates a STUN candidate, but it is eliminated because it is redundant with the host candidate, as noted in [RFC5245] Sec 4.1.2..

5. The application-provided TURN server as seen through the virtual interface. (Traffic through this candidate is recursively encapsulated.)

   * e.g. candidate:702786350 1 udp 24583935 [public ip addr of the application TURN server] 52631 typ relay raddr [public ip addr for the proxy] rport 54606 generation 0

There are no STUN or TURN candidates on the physical interfaces, because the application-specified STUN and TURN servers are not reachable through the firewall.

If the remote peer is within the same network, it may be possible to establish a direct connection using both peers' host candidates. If the network prevents this kind of direct connection, the path will instead take a "hairpin" route through the enterprise's proxy, using one peer's physical "host" candidate and the other's virtual "host" candidate, or (if that is also disallowed by the network configuration) a "double hairpin" using both endpoints' virtual "host" candidates.

6.2. Conflicting proxies configured by Auto-Discovery and local policy

Consider an enterprise network with TURN and HTTP proxies advertised for Auto-Discovery with unspecified leakiness (thus defaulting to leaky). The browser endpoint configures an additional TURN proxy by a proprietary local mechanism.

If the locally configured proxy is leaky, then the browser MUST produce candidates representing any physical interfaces (including SSLTCP routes through the HTTP proxy), plus candidates for both UDP-only virtual interfaces created by the two TURN servers.

There MUST NOT be any candidate that uses both proxies. Multiple configured proxies are not chained recursively.

If the locally configured proxy is "sealed", then the browser MUST produce only candidates from the virtual interface associated with that proxy.

If both proxies are configured for "sealed" use, then the browser MUST produce only candidates from the virtual interface associated with the proxy with higher rank.

7. Security Considerations

A RETURN proxy can capture, block, and otherwise interfere with all of its clients' WebRTC network activity. Therefore, browsers and other WebRTC endpoints MUST NOT use RETURN proxies that are provided by untrusted sources. For example, endpoints MUST NOT implement a configuration based on unauthenticated network multicast (e.g. mDNS) unless the endpoint will only be used on networks where all other users are fully trusted to intercept all WebRTC traffic. In contrast, endpoints MAY implement mechanisms to configure RETURN proxies by system-wide policy, which can only be modified by trusted system administrators.

This document describes web browser behaviors that, if implemented correctly, allow users to achieve greater identity-confidentiality during WebRTC calls under certain configurations.

If a site administrator offers the site's users a TURN proxy, websites running in the users' browsers will be able to initiate a UDP-based WebRTC connection to any UDP transport address via the proxy. Websites' connections will quickly terminate if the remote endpoint does not reply with a positive indication of ICE consent, but no such restriction applies to other applications that access the TURN server. Administrators should take care to provide TURN access credentials only to the users who are authorized to have global UDP network access.

TURN proxies and application TURN servers can provide some privacy protection by obscuring the identity of one peer from the other. However, unencrypted TURN provides no additional privacy from an observer who can monitor the link between the TURN client and server, and even encrypted TURN ([I-D.ietf-tram-stun-dtls] Section 4.6) does not provide significant privacy from an observer who sniff traffic on both legs of the TURN connection, due to packet timing correlations.

8. IANA Considerations

This document requires no actions from IANA.

9. Acknowledgements

Thanks to Harald Alvestrand, Philipp Hancke, Tirumaleswar Reddy, Alan Johnston, John Yoakum, and Cullen Jennings for suggestions to improve the content and presentation. Special thanks to Alan Johnston for contributing the visual overview in Section 2.

10. References

10.1. Normative References

[I-D.ietf-rtcweb-jsep]
Uberti, J. and C. Jennings, "Javascript Session Establishment Protocol", draft-ietf-rtcweb-jsep-06 (work in progress), February 2014.

[RFC1928] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and L. Jones, "SOCKS Protocol Version 5", RFC 5766, March 1996.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

[RFC6156] Camarillo, G., Novo, O., and S. Perreault, "Traversal Using Relays around NAT (TURN) Extension for IPv6", RFC 6156, April 2011.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, April 2012.

10.2. Informative References

[I-D.ietf-rtcweb-security]
Rescorla, E., "Security Considerations for WebRTC", ietf-rtcweb-security-07 (work in progress), July 2014.

[I-D.ietf-rtcweb-use-cases-and-requirements]
Holmberg, C., Hakansson, S., and G. Eriksson, "Web Real-Time Communication Use-cases and Requirements", ietf-rtcweb-use-cases-and-requirements-14 (work in progress), February 2014.

[I-D.ietf-tram-stun-dtls]
Petit-Huguenin, M. and G. Salgueiro, "Datagram Transport Layer Security (DTLS) as Transport for Session Traversal Utilities for NAT (STUN)", ietf-rtcweb-use-cases-and-requirements-14 (work in progress), June 2014.

[I-D.ietf-tram-turn-server-discovery]
Patil, P., Reddy, T., and D. Wing, "TURN Server Auto Discovery", draft-ietf-tram-turn-server-discovery-00 (work in progress), July 2014.

[I-D.reddy-mmusic-ice-best-interface-pcp]
Reddy, T., Wing, D., VerSteeg, B., Penno, R., and V. Singh, "Improving ICE Interface Selection Using Port Control Protocol (PCP) Flow Extension", draft-ietf-tram-turn-server-discovery-00 (work in progress), October 2013.

Authors' Addresses

Benjamin M. Schwartz
Google, Inc.
111 8th Ave
New York, NY 10011
USA

Email: bemasc@webrtc.org

Justin Uberti
Google, Inc.
747 6th Street South
Kirkland, WA 98033
USA

Email: justin@uberti.name

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 20, 2017

P. Jones
S. Dhesikan
C. Jennings
Cisco Systems
D. Druta
AT&T

August 19, 2016

DSCP Packet Markings for WebRTC QoS
draft-ietf-tsvwg-rtcweb-qos-18

Abstract

Many networks, such as service provider and enterprise networks, can provide different forwarding treatments for individual packets based on Differentiated Services Code Point (DSCP) values on a per-hop basis. This document provides the recommended DSCP values for web browsers to use for various classes of WebRTC traffic.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 20, 2017.

Table of Contents

1. Introduction

Differentiated Services Code Point (DSCP) [RFC2474] packet marking can help provide QoS in some environments. This specification provides default packet marking for browsers that support WebRTC applications, but does not change any advice or requirements in other IETF RFCs. The contents of this specification are intended to be a simple set of implementation recommendations based on the previous RFCs.

Networks where these DSCP markings are beneficial (likely to improve QoS for WebRTC traffic) include:

1. Private, wide-area networks. Network administrators have control over remarking packets and treatment of packets.

2. Residential Networks. If the congested link is the broadband uplink in a cable or DSL scenario, often residential routers/NAT support preferential treatment based on DSCP.

3. Wireless Networks. If the congested link is a local wireless network, marking may help.

There are cases where these DSCP markings do not help, but, aside from possible priority inversion for "less than best effort traffic"

(see Section 5), they seldom make things worse if packets are marked appropriately.

DSCP values are in principle site specific, with each site selecting its own code points for controlling per-hop-behavior to influence the QoS for transport-layer flows. However in the WebRTC use cases, the browsers need to set them to something when there is no site specific information. This document describes a subset of DSCP code point values drawn from existing RFCs and common usage for use with WebRTC applications. These code points are intended to be the default values used by a WebRTC application. While other values could be used, using a non-default value may result in unexpected per-hop behavior. It is RECOMMENDED that WebRTC applications use non-default values only in private networks that are configured to use different values.

This specification defines inputs that are provided by the WebRTC application hosted in the browser that aid the browser in determining how to set the various packet markings. The specification also defines the mapping from abstract QoS policies (flow type, priority level) to those packet markings.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The terms "browser" and "non-browser" are defined in [RFC7742] and carry the same meaning in this document.

3. Relation to Other Specifications

This document is a complement to [RFC7657], which describes the interaction between DSCP and real-time communications. That RFC covers the implications of using various DSCP values, particularly focusing on Real-time Transport Protocol (RTP) [RFC3550] streams that are multiplexed onto a single transport-layer flow.

There are a number of guidelines specified in [RFC7657] that apply to marking traffic sent by WebRTC applications, as it is common for multiple RTP streams to be multiplexed on the same transport-layer flow. Generally, the RTP streams would be marked with a value as appropriate from Table 1. A WebRTC application might also multiplex data channel [I-D.ietf-rtcweb-data-channel] traffic over the same 5-tuple as RTP streams, which would also be marked as per that table. The guidance in [RFC7657] says that all data channel traffic would be marked with a single value that is typically different than the

value(s) used for RTP streams multiplexed with the data channel traffic over the same 5-tuple, assuming RTP streams are marked with a value other than default forwarding (DF). This is expanded upon further in the next section.

This specification does not change or override the advice in any other IETF RFCs about setting packet markings. Rather, it simply selects a subset of DSCP values that is relevant in the WebRTC context.

The DSCP value set by the endpoint is not trusted by the network. In addition, the DSCP value may be remarked at any place in the network for a variety of reasons to any other DSCP value, including default forwarding (DF) value to provide basic best effort service. Even so, there is benefit in marking traffic even if it only benefits the first few hops. The implications are discussed in Secton 3.2 of [RFC7657]. Further, a mitigation for such action is through an authorization mechanism. Such an authorization mechanism is outside the scope of this document.

4. Inputs

WebRTC applications send and receive two types of flows of significance to this document:

- media flows which are RTP streams [I-D.ietf-rtcweb-rtp-usage]
- data flows which are data channels [I-D.ietf-rtcweb-data-channel]

Each of the RTP streams and distinct data channels consists of all of the packets associated with an independent media entity, so an RTP stream or distinct data channel is not always equivalent to a transport-layer flow defined by a 5-tuple (source address, destination address, source port, destination port, and protocol). There may be multiple RTP streams and data channels multiplexed over the same 5-tuple, with each having a different level of importance to the application and, therefore, potentially marked using different DSCP values than another RTP stream or data channel within the same transport-layer flow. (Note that there are restrictions with respect to marking different data channels carried within the same SCTP association as outlined in Section 5.)

The following are the inputs provided by the WebRTC application to the browser:

- Flow Type: The application provides this input because it knows if the flow is audio, interactive video [RFC4594] [G.1010] with or without audio, or data.

- Application Priority: Another input is the relative importance of an RTP stream or data channel. Many applications have multiple flows of the same Flow Type and often some flows are more important than others. For example, in a video conference where there are usually audio and video flows, the audio flow may be more important than the video flow. JavaScript applications can tell the browser whether a particular flow is high, medium, low or very low importance to the application.

[I-D.ietf-rtcweb-transports] defines in more detail what an individual flow is within the WebRTC context and priorities for media and data flows.

Currently in WebRTC, media sent over RTP is assumed to be interactive [I-D.ietf-rtcweb-transports] and browser APIs do not exist to allow an application to to differentiate between interactive and non-interactive video.

5. DSCP Mappings

The DSCP values for each flow type of interest to WebRTC based on application priority are shown in Table 1. These values are based on the framework and recommended values in [RFC4594]. A web browser SHOULD use these values to mark the appropriate media packets. More information on EF can be found in [RFC3246]. More information on AF can be found in [RFC2597]. DF is default forwarding which provides the basic best effort service [RFC2474].

WebRTC use of multiple DSCP values may encounter network blocking of packets with certain DSCP values. See section 4.2 of [I-D.ietf-rtcweb-transports] for further discussion, including how WebRTC implementations establish and maintain connectivity when such blocking is encountered.

Flow Type	Very Low	Low	Medium	High
Audio	CS1 (8)	DF (0)	EF (46)	EF (46)
Interactive Video with or without Audio	CS1 (8)	DF (0)	AF42, AF43 (36, 38)	AF41, AF42 (34, 36)
Non-Interactive Video with or without Audio	CS1 (8)	DF (0)	AF32, AF33 (28, 30)	AF31, AF32 (26, 28)
Data	CS1 (8)	DF (0)	AF11	AF21

Table 1: Recommended DSCP Values for WebRTC Applications

The application priority, indicated by the columns "very low", "low", "Medium", and "high", signifies the relative importance of the flow within the application. It is an input that the browser receives to assist in selecting the DSCP value and adjusting the network transport behavior.

The above table assumes that packets marked with CS1 are treated as "less than best effort", such as the LE behavior described in [RFC3662]. However, the treatment of CS1 is implementation dependent. If an implementation treats CS1 as other than "less than best effort", then the actual priority (or, more precisely, the per-hop-behavior) of the packets may be changed from what is intended. It is common for CS1 to be treated the same as DF, so applications and browsers using CS1 cannot assume that CS1 will be treated differently than DF [RFC7657]. However, it is also possible per [RFC2474] for CS1 traffic to be given better treatment than DF, thus caution should be exercised when electing to use CS1. This is one of the cases where marking packets using these recommendations can make things worse.

Implementers should also note that excess EF traffic is dropped. This could mean that a packet marked as EF may not get through, although the same packet marked with a different DSCP value would have gotten through. This is not a flaw, but how excess EF traffic is intended to be treated.

The browser SHOULD first select the flow type of the flow. Within the flow type, the relative importance of the flow SHOULD be used to select the appropriate DSCP value.

Currently, all WebRTC video is assumed to be interactive [I-D.ietf-rtcweb-transports], for which the Interactive Video DSCP values in Table 1 SHOULD be used. Browsers MUST NOT use the AF3x DSCP values (for Non-Interactive Video in Table 1) for WebRTC applications. Non-browser implementations of WebRTC MAY use the AF3x DSCP values for video that is known not to be interactive, e.g., all video in a WebRTC video playback application that is not implemented in a browser.

The combination of flow type and application priority provides specificity and helps in selecting the right DSCP value for the flow. All packets within a flow SHOULD have the same application priority. In some cases, the selected application priority cell may have multiple DSCP values, such as AF41 and AF42. These offer different drop precedences. The different drop precedence values provides additional granularity in classifying packets within a flow. For example, in a video conference the video flow may have medium application priority, thus either AF42 or AF43 may be selected. More important video packets (e.g., a video picture or frame encoded without any dependency on any prior pictures or frames) might be marked with AF42 and less important packets (e.g., a video picture or frame encoded based on the content of one or more prior pictures or frames) might be marked with AF43 (e.g., receipt of the more important packets enables a video renderer to continue after one or more packets are lost).

It is worth noting that the application priority is utilized by the coupled congestion control mechanism for media flows per [I-D.ietf-rmcat-coupled-cc] and the SCTP scheduler for data channel traffic per [I-D.ietf-rtcweb-data-channel].

For reasons discussed in Section 6 of [RFC7657], if multiple flows are multiplexed using a reliable transport (e.g., TCP) then all of the packets for all flows multiplexed over that transport-layer flow MUST be marked using the same DSCP value. Likewise, all WebRTC data channel packets transmitted over an SCTP association MUST be marked using the same DSCP value, regardless of how many data channels (streams) exist or what kind of traffic is carried over the various SCTP streams. In the event that the browser wishes to change the DSCP value in use for an SCTP association, it MUST reset the SCTP congestion controller after changing values. Frequent changes in the DSCP value used for an SCTP association are discouraged, though, as this would defeat any attempts at effectively managing congestion. It should also be noted that any change in DSCP value that results in a reset of the congestion controller puts the SCTP association back into slow start, which may have undesirable effects on application performance.

For the data channel traffic multiplexed over an SCTP association, it is RECOMMENDED that the DSCP value selected be the one associated with the highest priority requested for all data channels multiplexed over the SCTP association. Likewise, when multiplexing multiple flows over a TCP connection, the DCSP value selected should be the one associated with the highest priority requested for all multiplexed flows.

If a packet enters a network that has no support for a flow type-application priority combination specified in Table 1, then the network node at the edge will remark the DSCP value based on policies. This could result in the flow not getting the network treatment it expects based on the original DSCP value in the packet. Subsequently, if the packet enters a network that supports a larger number of these combinations, there may not be sufficient information in the packet to restore the original markings. Mechanisms for restoring such original DSCP is outside the scope of this document.

In summary, DSCP marking provides neither guarantees nor promised levels of service. However, DSCP marking is expected to provide a statistical improvement in real-time service as a whole. The service provided to a packet is dependent upon the network design along the path, as well as the network conditions at every hop.

6. Security Considerations

Since the JavaScript application specifies the flow type and application priority that determine the media flow DSCP values used by the browser, the browser could consider application use of a large number of higher priority flows to be suspicious. If the server hosting the JavaScript application is compromised, many browsers within the network might simultaneously transmit flows with the same DSCP marking. The DiffServ architecture requires ingress traffic conditioning for reasons that include protecting the network from this sort of attack.

Otherwise, this specification does not add any additional security implications beyond those addressed in the following DSCP-related specifications. For security implications on use of DSCP, please refer to Section 7 of [RFC7657] and Section 6 of [RFC4594]. Please also see [I-D.ietf-rtcweb-security] as an additional reference.

7. IANA Considerations

This specification does not require any actions from IANA.

8. Downward References

This specification contains a downwards reference to [RFC4594] and [RFC7657]. However, the parts of the former RFC used by this specification are sufficiently stable for this downward reference. The guidance in the latter RFC is necessary to understand the Diffserv technology used in this document and the motivation for the recommended DSCP values and procedures.

9. Acknowledgements

Thanks to David Black, Magnus Westerlund, Paolo Severini, Jim Hasselbrook, Joe Marcus, Erik Nordmark, Michael Tuexen, and Brian Carpenter for their invaluable input.

10. Dedication

This document is dedicated to the memory of James Polk, a long-time friend and colleague. James made important contributions to this specification, including serving initially as one of the primary authors. The IETF global community mourns his loss and he will be missed dearly.

11. Document History

Note to RFC Editor: Please remove this section.

This document was originally an individual submission in RTCWeb WG. The RTCWeb working group selected it to be become a WG document. Later the transport ADs requested that this be moved to the TSVWG WG as that seemed to be a better match.

12. References

12.1. Normative References

[I-D.ietf-rtcweb-data-channel]
Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", draft-ietf-rtcweb-data-channel-13 (work in progress), January 2015.

[I-D.ietf-rtcweb-rtp-usage]
Perkins, D., Westerlund, M., and J. Ott, "Web Real-Time Communication (WebRTC): Media Transport and Use of RTP", draft-ietf-rtcweb-rtp-usage-26 (work in progress), March 2016.

[I-D.ietf-rtcweb-security]
Rescorla, E., "Security Considerations for WebRTC", draft-ietf-rtcweb-security-08 (work in progress), February 2015.

[I-D.ietf-rtcweb-transports]
Alvestrand, H., "Transports for WebRTC", draft-ietf-rtcweb-transports-15 (work in progress), August 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, DOI 10.17487/RFC4594, August 2006, <http://www.rfc-editor.org/info/rfc4594>.

[RFC7657] Black, D., Ed. and P. Jones, "Differentiated Services (Diffserv) and Real-Time Communication", RFC 7657, DOI 10.17487/RFC7657, November 2015, <http://www.rfc-editor.org/info/rfc7657>.

[RFC7742] Roach, A., "WebRTC Video Processing and Codec Requirements", RFC 7742, DOI 10.17487/RFC7742, March 2016, <http://www.rfc-editor.org/info/rfc7742>.

12.2. Informative References

[G.1010] International Telecommunications Union, "End-user multimedia QoS categories", Recommendation ITU-T G.1010, November 2001.

[I-D.ietf-rmcat-coupled-cc]
Islam, S., Welzl, M., and S. Gjessing, "Coupled congestion control for RTP media", draft-ietf-rmcat-coupled-cc-03 (work in progress), July 2016.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <http://www.rfc-editor.org/info/rfc2474>.

[RFC2597] Heinanen, J., Baker, F., Weiss, W., and J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, DOI 10.17487/RFC2597, June 1999, <http://www.rfc-editor.org/info/rfc2597>.

[RFC3246] Davie, B., Charny, A., Bennet, J., Benson, K., Le Boudec, J., Courtney, W., Davari, S., Firoiu, V., and D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)", RFC 3246, DOI 10.17487/RFC3246, March 2002, <http://www.rfc-editor.org/info/rfc3246>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <http://www.rfc-editor.org/info/rfc3550>.

[RFC3662] Bless, R., Nichols, K., and K. Wehrle, "A Lower Effort Per-Domain Behavior (PDB) for Differentiated Services", RFC 3662, DOI 10.17487/RFC3662, December 2003, <http://www.rfc-editor.org/info/rfc3662>.

Authors' Addresses

Paul E. Jones
Cisco Systems

Email: paulej@packetizer.com

Subha Dhesikan
Cisco Systems

Email: sdhesika@cisco.com

Cullen Jennings
Cisco Systems

Email: fluffy@cisco.com

Dan Druta
AT&T

Email: dd5826@att.com

rtcweb
Internet-Draft
Intended status: Informational
Expires: January 9, 2017

P. Patel
C. Jennings
S. Nandakumar
J. Rosenberg
D. Wing
Cisco

July 08, 2016

Firewall Traversal for WebRTC
draft-jennings-behave-rtcweb-firewall-05

Abstract

Traversal of RTP through corporate firewalls has traditionally been complex, requiring the deployment of Session Border Controllers (SBCs) or wide open pinholes. This draft proposes a simple technique that allows WebRTC based RTP traffic to traverse firewalls without complex firewall configuration and without deployment of SBCs or other middleboxes.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 9, 2017.

Copyright Notice

Table of Contents

1. Problem Statement

WebRTC [I-D.ietf-rtcweb-overview] based voice and video communications systems are becoming far more common inside enterprises, which often need voice and video media to traverse the enterprise firewall. This can happen when a device inside the firewall such as a web browser or phone is exchanging media with a conference bridge or gateway outside the firewall, or it can happen when a device inside the firewall is talking to a device in another enterprise or behind a different firewall.

This problem is not unique to WebRTC media of course. It is common practice for enterprise administrators to block outbound UDP through the corporate firewall. This is done for several reasons:

1. The lack of any kind of return messages means that there is no way to know that the recipient of the UDP traffic really wants it. Infected computers within the enterprise could utilize UDP as the source of a DDoS attack. If the firewall permitted such outbound traffic, the enterprise could in effect be a contributing source to such an attack. By blocking UDP, the enterprise IT admin ensures that this cannot happen - at least not to external targets.

2. There have been prior attacks that have utilized UDP as a command and control channel for orchestrating DDoS attacks. At the time, UDP had little usage within enterprises (most VoIP was internal to the enterprise when it existed at all). Consequently, infosec departments have deemed it safer to block UDP outright in order to prevent such further incidents.

3. Many IT administrators enable various packet inspection operations on traffic flowing through the firewall. High volume UDP traffic - such as voice or video - can be costly to inspect. As such, in cases where there is a need for traversal of such traffic, IT has preferred to deploy an SBC that, in essence, verifies that the traffic is VoIP and authorizes its egress. The IT administrator then enables traffic to/from the SBC through the firewall. In other words, VoIP authorization is delegated to an outsourced SBC.

As more and more IP communications services move to the cloud, there is an increased need for VoIP traffic to traverse the enterprise firewall. At the same time, the entire point of a cloud service is that it does not require the deployment of on premises infrastructure, making SBC-based solutions less desirable. An alternative solution that has been historically used is to enable outbound UDP in the firewall to specific IP addresses, corresponding to the external service (TURN servers or conference servers) that the enterprise wishes to authorize. With more applications running on virtual machines within cloud compute platforms like Amazon EC2, IP addresses are decreasingly usable as identifiers for a service. VMs running TURN servers or conferencing servers may be established and torn down by the day, hour or even minute, with continuously changing IP addresses. Given the multitenant nature of such providers, IT departments are unwilling to whitelist the IP addresses for the entire block used by such providers.

Consequently, there is a growing need for solutions that allow VoIP traversal through the corporate firewall that alleviate the concerns above. This issue is further exacerbated by the growing adoption of WebRTC by enterprise applications, which provide a ready source of RTP traffic which often needs to traverse the firewall.

2. Solution Requirements

We believe the solution must meet the following requirements:

REQ-1: The solution must enable traversal of real-time media without requiring deployment of additional media intermediaries on premise (e.g., no SBC required)

REQ-2: The solution must not require the whitelisting of specific external IP addresses

REQ-3: The solution must enable the enterprise to be sure that the receiving party of the traffic desires the traffic

REQ-4: The solution must work with P2P calls between users in different enterprises without requiring a TURN server

REQ-5: The solution must work with cloud services external to the enterprise which terminate media on servers, such as conference servers, voicemail servers, and so on.

REQ-6: The solution must not require decryption of either signaling or media traffic at the firewall or at any other intermediary

REQ-7: The solution must allow the IT department to easily make policy decisions about which applications are allowed, or not allowed, to traverse the firewall

REQ-8: The solution must not require inspection of every single packet that traverses the firewall

REQ-9: The solution must provide a minimum level of proof that the traffic is WebRTC media or data and not something else

REQ-10: The solution must work with WebRTC traffic. Note that solving this for non-WebRTC is a non-requirement.

3. Solution Overview

Many of the reasons for blocking UDP at the corporate firewall have their origins in the lack of a three-way handshake for UDP traffic. TCP’s three-way handshake ensures that the receiving party of the

connection desires the traffic. Similarly, HTTP traffic easily traverses the firewall since it provides application identification information in the URL.

Consequently, the solution proposed here relies on the ICE connectivity checks, which provide a similar handshake and ensure consent of the remote party.

The firewall looks for an initial STUN transaction to a STUN server to learn which application is using the port (based on the STUN HOST attribute Section 6). Next the firewall watches the outbound ICE connectivity check on that port and allows inbound ICE connectivity checks that are going to the same location that sent the outbound request and that have the correct random ufrag value that was created by the client inside the firewall. After a successful ICE connectivity check, the firewall allows other media to flow on the same 5 tuple that had the successful ICE connectivity check. Timers are used to removed the various pinholes created.

In addition, the initial outbound STUN packets can contain the STUN HOST attribute which the firewall can use to make an authorization decision on the application.

The end result is a system where:

- STUN packets are only allowed "in" if they know the crypto random username generated by a client inside the firewall
- STUN packers going "out" can be restricted by policy based on the hostname of the STUN server they are using
- Non STUN packets are only allowed "in" if they match a 5 tuple that a client inside the firewall sent a packet too
- Non STUN packets are only allowed "out" if the destination they are sending to did a stun consent handshake

4. Firewall Processing

The firewall processing is broken into four stages: recognizing STUN packets, mapping to an application, making a policy decision as to whether each STUN packet should trigger a pinhole to be created, and managing the lifetime of any pinholes that are created.

4.1. Terminology

The key words defined in [RFC2119] are used in this specification.

The term 3-tuple is used to refer to IP address, protocol (which is always UDP), and port that the firewall sees as the address of the client inside the firewall.

The term 4-tuple is used to refer to 3-tuple plus the ice ufrag that was send in the STUN request message for the client inside the firewall.

The term 5-tuple is used to refer to the 3-tuple plus the IP address and port of the device outside the firewall.

When matching a ufrag, if it is a STUN request that came from outside the firewall, the two halves of the username on either side of the ":" need to be swapped before matching.

4.2. Recognizing STUN packets

STUN messages all have a magic cookie value of 0x2112A442 in the 4th to 8th byte. This can be used to quickly filter nearly all UDP packets that are not STUN packets. Many firewalls are capable of doing this in hardware. STUN supports an optional FINGERPRINT attribute that provides a 32 bit CRC over the message.

Option A: Firewalls SHOULD look at outbound UDP packets and if they have the correct magic cookie they can classify them as STUN packets.

Option B: The firewall looks for any outgoing STUN requests to the STUN port (3478). When it finds one, it stores the 3 tuple of the source address port and protocol=UDP and for the next 30 seconds checks any packets from this 3 tuple to see if they are ICE connectivity checks.

Open Issues:

- decide between option A and B. A requires looking at all UDP packets but will likely work better than B. Most firewalls look at all TCP packets so probably not too big of a deal.

- MAY, MUST, MUST NOT look at FINGERPRINT - what do we want here. If we put MAY or MUST, then browsers MUST include this. If browsers are not required to provide this then I think we are more in the MUST NOT category. If we do not use the fingerprint, there will be some small number of false positives.

- Should do the analysis to see what harm comes of treating random packets as STUN packets.

- CJ Proposal: Browsers MUST send the fingerprint when sending STUN messages to STUN server but MAY use it when doing ICE connectivity checks. This is to help save bandwidth as Justin Uberti was suggesting that change from approximately 50kbps to 100 kbps for ICE makes big difference for mobile devices.

4.3. Application Mapping

The STUN HOST attribute Section 6 carries the fully qualified domain name of the STUN server that is being contacted in the STUN requests. So for example, if a browser was on a page such as example.com and that page used the WebRTC calls to set up a connection to stun.example.com, the STUN request's HOST attribute would have the value stun.example.com. Similarly when contacting a STUN or TURN server over TLS or DTLS, the TLS SNI [RFC6066] value provides the name of the host. For systems that provide a unique STUN server name for each application, this allows the firewall to map the stun port to the application using it and use that for logging and making any policy decisions.

Once the Firewall receives as STUN packet from the inside to the outside on a new 3-tuple. It MUST create an internal record to track any additional traffic on this 3-tuple. If the STUN packets contains an HOST attribute then the value it contains is saved in this record and referred to as the applications name. Firewall might wish to put the application name in the log files for this 3-tuple.

It is important to realize that any application inside the firewall can lie about the value of the HOST attribute. However, a web browser that is trusted will not allow the Javascript running in the web browser to lie about the value of the HOST.

4.4. Policy decision

Once the firewall has received a STUN packet from inside the firewall, it needs to decide if the packet is acceptable. For most situations the firewall SHOULD accept all outbound STUN packets. This is similar to allowing all outbound TCP flows. Some firewalls may choose to look at other factors including the outside UDP port and the application name for this 3-tuple.

In general WebRTC media can be sent on a wide range of UDP ports but the two ports that are commonly used are the the RTP port (5004) and TURN port (3478). Some firewalls MAY choose to only allow flows

where the destination port on the outside of the firewall is one of these.

Some firewalls MAY decide to white or blacklist connections based on the application name.

4.5. Creating the pinhole rules

Once a STUN packet is accepted, the firewall MUST create a temporary rule that causes the firewall to allow any inbound or outbound ICE messages on this 4-tuple. This pinhole MUST to be valid for at least 5 seconds from the time of creation.

The firewall keeps track of the STUN transaction ID for all STUN requests messages that traverse the 4 tuple along with the 5 tuple they were sent on and direction (inbound or outbound). If the firewall sees a STUN Success binding responses, with the matching transaction ID, and on the same 5 tuple but in the opposite direction as the STUN request, then a valid ICE connectivity check has happened. Then the firewall MUST create a pinhole for this 5 tuple that allows any UDP traffic to flow across that 5 tuple. This pinhole MUST to be valid for at least 30 seconds from the time of creation.

The firewall continues watching ICE connectivity checks across this 5-tuple as described in the previous paragraph and anytime the a valid ICE connectivity check happens, this effectively extends the lifetime of the pinhole by 30 seconds. The procedures in [RFC7675] will ensure that an ICE connectivity check is done more often than every 30 seconds. This is designed to make things work with behave compliant NATs and Firewalls as specified in [RFC4787].

4.6. Media vs Data Statistics

WebRTC can send audio and video as well as carry a data channel. Confidential data could leave an enterprise by a video camera being pointed at a document, but IT departments are often more concerned about the data channel. It is easy for the firewall to separately track the amount of RTP media and non-media data for each WebRTC flow. If the first byte of the UDP message is 23, it is non-media data; if it is in the range 127 to 192 it is audio or video data. More information about this can be found in [I-D.ietf-avtcore-rfc5764-mux-fixes]. Network management systems on the firewall can track these two separately which can help identify unusual usage.

5. WebRTC Browsers

This specification would require browsers to include the FINGERPRINT and the HOST attributes in STUN requests to a STUN server used to gather candidates for this to work correctly. Note they MUST not be included in STUN requests sent peer to peer or sent to ensure media consent.

Open Issue: how much randomness for ICE ufrag

- ICE mandates at least 24 bits of randomness but we could require the browsers produce 64 bits of randomness?

Open Issue: Does adding the HOST reduce user privacy?

- Consider the following case. The user goes to https://facebook.com and initiates a call with another Facebook user using facebook.com as the name of the STUN server. The domain facebook.com will appear (unencrypted) in the STUN packets sent from the browser to Facebook's TURN server. Anyone along the network path could tell that the user is using Facebook's TURN server. However, when the original TLS connection for the HTTP was made, the Server Name Indication (SNI) in the TLS of the HTTPS connection also revealed facebook.com, largely for the same reasons - so that the firewall would be able to see which applications are using the network.

Open Issue: Would only including HOST when it matched the HTTP ORIGIN improve privacy?

- We could make this so that when used with WebRTC browsers, the HOST is only included in the STUN messages when the name of the STUN servers matches the HTTP ORIGIN of the web page initiating the STUN request. It is not clear if this would improve privacy or not.

6. STUN HOST attribute

This specification defines a new STUN attribute called HOST and uses the syntax defined in Section 15 of [RFC5389]. This attribute is of type comprehension-optional. The value of the HOST attribute is a variable length value. It MUST contain a UTF-8 [RFC3629] encoded sequence of characters.

The HOST attribute identifies the fully qualified domain name of the application provider that is serving the WebRTC application and also operating the STUN server. The WebRTC EndPoint MUST include this

attribute as part of the ICE candidate gathering phase and there MUST be only one HOST attribute in a given STUN binding request.

7. Deployment Advice

7.1. WebRTC Servers

WebRTC media servers and TURN servers with public IP address(es) that can receive incoming packets from anywhere on the Internet are suggested to listen for UDP on ports 5004 for RTP media servers and 3478 for TURN servers. UDP destined for port 53 or 123 if often allowed by firewalls that otherwise block UDP.

7.2. Firewall Admins

Often the approach has been to lock down everything, so that all UDP is blocked. This simply causes applications to do things like embed the data in normal looking HTTP or HTTPS requests. Malware and viruses use similar approaches. Just turning off all UDP results in a poor user experience some of the time, which results in users moving to applications and devices outside the firewall. The IT department loses the visibility into what is going on and can no longer protect its users when their computers become compromised. Allowing things that users want to use to work and monitoring them to detect when things have gone wrong is very valuable.

8. Design Consideration

8.1. Why not just use TCP?

TODO

9. IANA Considerations

IANA is requested to add the HOST attribute to the STUN attribute registry. The values for HOST is to be allocated from the expert review comprehension-optional range of (0xC000 - 0xFFFF).

Value	Name	Reference
TBD	HOST	RFCXXXX

This specification defines the HOST attribute in Section 6.

10. Security Considerations

Enterprises have a range of concerns around WebRTC traffic traversal of the firewall. The major concerns that are raised include:

1. Unlike TCP, UDP does not have a connection where a device inside the firewall has confirmed that it wants to talk to the thing outside.

2. Incoming UDP pinholes allow out of band packets to be spoofed into connecting as there is no equivalent of a TCP sequence number to check.

3. UDP has been used by malware command and control protocols so we block it.

4. We do not want enable ways for data to be exfiltrated outside the firewall with no monitoring.

5. An encrypted data channel in WebRTC can be used to bring malware into the company.

6. An encrypted media or data channel in WebRTC can be used as a command and control channel for malware inside the firewall.

7. An encrypted data channel in WebRTC can be used by an outside attacker to exfiltrate private files from inside the firewall.

TODO - Describe to what degree theses are addressed. Be clear about attacks due to Javascript inside the firewall and attacks due to executables inside the firewall.

10.1. Privacy

Unlike previous version of this draft, we think that using HOST instead of ORIGIN minimizes any privacy concerns. The HOST is already known to the operator of the STUN server as they run it. It often contains exactly the same information as the existing STUN REALM attribute. It has roughly the same information as the TLS SNI [RFC6066] when STUN is run over DTLS.

11. Alternate Approaches

11.1. SDN Control of Firewall

An alternative ways of solving this problem is for the Web Application running in the browser to inform the web site what ports and IP addresses it is using then the web site to contact the

appropriate SDN controller and request the SDN controller tell the appropriate firewall what pinholes to open. This can be made to work in some deployments but not all as it is often not clear how to find the correct SDN controller or set up a relationship such that the SDN controller trusts a website outside the firewall wall enough to let it tell the controller to open wholes in the Firewall.

SDN based approaches should be pursued as well as this approach as they compliment each other.

11.2. Any Cast White List

Deploying media or TURN servers on a single any-cast IP address also makes it easier for firewall administrators to whitelist the address. Concerns have been raised that two packets sent from the same host to a given any-cast address may get delivered to different servers. This is certainly possible in theory but in practice it does not seem be happen in limited experiments done so far.

12. Acknowledgements

Many thanks to review from Shaun Cooley, Teh Cheng, and Alissa Cooper.

The definition of HOST STUN attribute was motivated by discussion around the draft-ietf-tram-stun-origin document and we want to thank Alan Johnston, Justin Uberti, John Yoakum, and Kundan Singh.

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <http://www.rfc-editor.org/info/rfc3629>.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, DOI 10.17487/RFC5389, October 2008, <http://www.rfc-editor.org/info/rfc5389>.

13.2. Informative References

[I-D.ietf-avtcore-rfc5764-mux-fixes]
Petit-Huguenin, M. and G. Salgueiro, "Multiplexing Scheme Updates for Secure Real-time Transport Protocol (SRTP) Extension for Datagram Transport Layer Security (DTLS)", draft-ietf-avtcore-rfc5764-mux-fixes-10 (work in progress), July 2016.

[I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-15 (work in progress), January 2016.

[RFC4787] Audet, F., Ed. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January 2007, <http://www.rfc-editor.org/info/rfc4787>.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <http://www.rfc-editor.org/info/rfc6066>.

[RFC7675] Perumal, M., Wing, D., Ravindranath, R., Reddy, T., and M. Thomson, "Session Traversal Utilities for NAT (STUN) Usage for Consent Freshness", RFC 7675, DOI 10.17487/RFC7675, October 2015, <http://www.rfc-editor.org/info/rfc7675>.

Authors' Addresses

Pradeep Patel
Cisco

Email: pradpate@cisco.com

Cullen Jennings
Cisco

Email: fluffy@iii.ca

Suhas Nandakumar
Cisco

Email: snandaku@cisco.com

Jonathan Rosenberg
Cisco

Email: jdrosen@cisco.com

Dan Wing
Cisco

Email: dwing@cisco.com

Rtcweb Working Group
Internet-Draft
Intended status: Informational
Expires: December 18, 2015

G. Mandyam
Qualcomm Innovation Center
M. Luby
Qualcomm Technologies Inc.
T. Stockhammer
Qualcomm CDMA Technologies GMBH

June 16, 2015

FEC FRAME Raptor Extensions for Multiple RTP Synchronization Sources
draft-mandyam-rtcweb-fecframebundledssrc-00

Abstract

The FEC FRAME Raptor code options do not currently address the case of bundled protection of multiple media types over multiple real-time transport protocol (RTP) synchronization sources (SSRC's). This document provides the FEC source and repair payload definitions that enable a single repair flow to be defined for multiple RTP flows

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 18, 2015.

Copyright Notice

Table of Contents

1. Introduction

[RFC6681] provides the specification of the fully formed Forward Error Correction (FEC) scheme for Raptor/RaptorQ codes in the context of the FEC Framework (FEC Frame - see [RFC6363]). This document provides extensions that allow for protection of multiple RTP flows where each flow has its own unique sequence number space. There are two approaches described: one using explicit source FEC payload ID's, and one that does not.

2. Multi-sequenced Flows with Explicit Source FEC Payload ID

As per Section 6 of [RFC6681], arbitrary flows (including RTP flows) can be protected if the source is identified explicitly using a Source FEC Payload ID. However, the Source FEC Payload ID must be sent along with the source payload to the receiver.

2.1. RTP Header Extension for Source FEC Payload ID

It is recommended that the Source FEC Payload ID as defined in Section 6.2.2 of [RFC6681] be used in an RTP header extension for each RTP source stream packet. Since the Source FEC Payload ID is 32 bits long (4 bytes), the 1-byte header extension solution in Section 4.2 of [RFC5285] is sufficient for identifying the Source FEC Payload ID. Note however that there may be reasons to use the 2-byte header extension solution provided in Section 4.3 of [RFC5285] (e.g. due to the need for 8-bit extension ID encoding).

2.2. Repair FEC Payload ID

The Repair FEC Payload ID is used as defined in Section 6.2.3 of [RFC6681]. This will be sent along with the associated repair payload in a repair FEC stream (i.e. RTP flow). This can also be sent as a RTP header extension (although it can be included in the RTP payload of the repair FEC stream). As with the Source FEC Payload ID, the 1-byte header extension method is preferred.

2.3. New RTP Header Extension URI's

[RFC EDITOR NOTE: Please replace RFCXXXX with the RFC number of this document.]

This document defines two new extension URI's in the RTP Compact Header Extensions subregistry of the Real-Time Transport Protocol (RTP) Parameters registry, according to the following data:

```
Extension URI: urn:ietf:params:rtp-hdrext:FEC-FR:SourceID
Description:   Source FEC Payload ID
Contact:       mandyam@quicinc.com
Reference:     RFCXXXX

Extension URI: urn:ietf:params:rtp-hdrext:FEC-FR:RepairID
Description:   Repair FEC Payload ID
Contact:       mandyam@quicinc.com
Reference:     RFCXXXX
```

3. Multi-sequenced Flows without Source FEC Payload ID

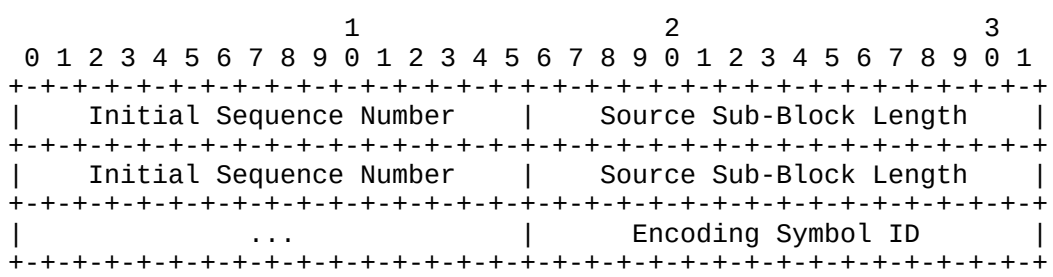
Section 8 of [RFC6681] describes the necessary procedures for single-sequenced flows. This section extends this method for multi-sequenced flows, in particular multiple RTP flows corresponding to different SSRC's. The FEC Scheme ID's used are 5 and 6.

3.1. Source FEC Payload ID

As with the approach provided in [RFC6681] for single-sequenced flows, a source FEC Payload ID is not used as the source packets are not modified.

3.2. Repair FEC Payload ID

In contrast to Section 8.1.3 of [RFC6681], only one format for the Repair FEC Payload is provided (based on Format A), but with necessary extensions for multi-sequenced flows. The number of flows in a repair packet and the order in which the flows appear in the repair packet are determined using out-of-band signalling (for an SDP example, see Section 5.2).



Repair FEC Payload ID (multisequence)

Figure 1

Initial Sequence Number (Flow i ISN), (16 bits): This field specifies the lowest 16 bits of the sequence number of the first packet to be included in this sub-block. If the sequence numbers are shorter than 16 bits, then the received Sequence Number SHALL be logically padded with zero bits to become 16 bits in length, respectively. The field type is unsigned integer.

Source Sub-Block Length (SSBL), (16 bits): This field specifies the length of the source sub-block in symbols. The field type is unsigned integer.

Encoding Symbol ID (ESI), (16 bits): This field indicates which repair symbols are contained within this repair packet. The ESI provided is the ESI of the first repair symbol in the packet. The field type is unsigned integer.

3.3. Procedures

There are slight changes necessary to the procedures outlined in Section 8.2 of [RFC6681] in order to accomodate multiple sequenced flows.

3.3.1. Source Symbol Contruction

FEC Scheme 5 and FEC Scheme 6 use the procedures defined in Section 5 of [RFC6681] to construct a set of source symbols to which the FEC code can be applied.

During the construction of the source block:

- the flow identifier, f[i], for each flow included in the source packet information.
- the length indication, l[i], included in the Source Packet Information for each packet shall be dependent on the protocol carried within the transport payload. Rules for RTP are specified below.
- the value of s[i] in the construction of the Source Packet Information for each packet shall be the smallest integer such that s[i]*T >= (l[i]+3).

3.3.2. Derivations of Source FEC Packet Identification Information

The Source FEC Packet Identification Information for a source packet is derived from the flows in each packet, sequence number of each individual flow of the packet, and information received in any repair FEC packet belonging to this source block. The application data units (ADU’s) that constitute the source block are identified by the associated flow identifier and sequence number of the first source packet in the block. This information is signaled in all repair FEC packets associated with the source block in the Initial Sequence Number field.

The length of the Source Packet Information (in octets) for source packets within a source block is equal to the length of the payload containing encoding symbols of the repair packets (i.e., not including the Repair FEC Payload ID) for that block, which MUST be the same for all repair packets. The Application Data Unit Information Length (ADUIL) in symbols is equal to this length divided by the encoding symbol size (which is signaled in the FEC Framework Configuration Information). The set of source packets included in the source block is determined by the Initial Sequence Number (ISN) and Source Sub-Block Length (SSBL) as follows:

Let,

- o f be the index of the flow, i.e. if f refers to the first flow in the source block then f=1.
- o I(f) be the Initial Sequence Number of the source sub-block from flow f
- o LP(f) be the Source Sub-Block Information Length in symbols for flow f
- o LB(f) be the Source Sub-Block Length in symbols for flow f

Then, source packets with sequence numbers from I(f) to I(f) +(LB(f)/LP(f))-1 for flow f inclusive are included in the source block. The Source Sub-Block Length, LB(f), MUST be chosen such that it is at least as large as the largest Source Packet Information Length LP(f).

Note that if no FEC repair packets are received, then no FEC decoding is possible, and it is unnecessary for the receiver to identify the Source FEC Packet Identification Information for the source packets.

For FEC Scheme 1, the ESI value placed into a repair packet is calculated as specified in Section 5.3.2 of [RFC5053].

For FEC Scheme 2, the ESI value placed into a repair packet is calculated as specified in Section 4.4.2 of [RFC6330].

In both cases, K is identical to the sum of all the SSBL’s indicated in the repair packet.

3.3.3. Procedures for RTP Source Flows

In the specific case of RTP source packet flows, the RTP Sequence Number field SHALL be used as the sequence number in the procedures described above. The length indication included in the Application Data Unit Information SHALL be the sum over all flows of the RTP payload length plus the length of the contributing sources (CSRCs), if any, the RTP Header Extension, if present, and the RTP padding octets, if any. Note that this length is always equal to the UDP payload length of the packet minus 12.

4. Registration of the ’bundled/raptorfec’ Media Type

This RTP payload format is identified using the ’bundled/raptorfec’ media type that is registered in accordance with [RFC4855] and uses

the template of [RFC4288]. The Media Type Definition is identical to 'video/raptorfec' and can be found in Section 6.2.1 of [RFC6682].

5. SDP Example

5.1. With RTP Extensions

An SDP example employing bundled protection of a video and audio stream (derived from Section 10 of [RFC6681]) is shown below. In this example, the SDP guidance provided in Section 5 of [RFC5285] is also used.

```
v=0
o=ali 1122334455 1122334466 IN IP4 fec.example.com
s=Raptor FEC Example
t=0 0
a=group:FEC-FR S1 S2 R1
m=video 30000 RTP/AVP 100
c=IN IP4 233.252.0.1/127
a=rtpmap:100 MP2T/90000
a=fec-source-flow: id=0
a=mid:S1
a=extmap:1 urn:ietf:params:rtp-hdrext:FEC-FR:SourceID
m=audio 10000 RTP/AVP 0 8 97
c=IN IP4 233.252.0.2/127
b=AS:200
a=rtpmap:0 PCMU/8000
a=mid:S2
a=extmap:1 urn:ietf:params:rtp-hdrext:FEC-FR:SourceID
a=fec-source-flow: id=1
m=application 30000 UDP/FEC
c=IN IP4 233.252.0.3/127
a=fec-repair-flow: encoding-id=6; fssi=Kmax:8192,T:128,P:A
a=repair-window:200ms
a=mid:R1
a=extmap:1 urn:ietf:params:rtp-hdrext:FEC-FR:RepairID
```

5.2. Without RTP Extensions

An SDP example employing bundled protection of a video and audio stream (derived from Section 10 of [RFC6681]) is shown below. In this example, source flows (S1 and S2) identified in the 'a=group' attirbute appear in this order in the Repair FEC Payload ID (see Figure 1).

```
v=0
o=ali 1122334455 1122334466 IN IP4 fec.example.com
s=Raptor FEC Example
t=0 0
a=group:FEC-FR S1 S2 R1
m=video 30000 RTP/AVP 100
c=IN IP4 233.252.0.1/127
a=rtpmap:100 MP2T/90000
a=fec-source-flow: id=0
a=mid:S1
m=audio 10000 RTP/AVP 0 8 97
c=IN IP4 233.252.0.2/127
b=AS:200
a=rtpmap:0 PCMU/8000
a=mid:S2
a=fec-source-flow: id=1
m=application 30000 UDP/FEC
c=IN IP4 233.252.0.3/127
a=fec-repair-flow: encoding-id=6; fssi=Kmax:8192,T:128,P:A
a=repair-window:200ms
a=mid:R1
```

6. IANA Considerations

This memo includes no request to IANA.

7. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4288] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", RFC 4288, December 2005.

[RFC4855] Casner, S., "Media Type Registration of RTP Payload Formats", RFC 4855, February 2007.

[RFC5052] Watson, M., Luby, M., and L. Vicisano, "Forward Error Correction (FEC) Building Block", RFC 5052, August 2007.

[RFC5053] Luby, M., Shokrollahi, A., Watson, M., and T. Stockhammer, "Raptor Forward Error Correction Scheme for Object Delivery", RFC 5053, October 2007.

[RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.

[RFC6330] Luby, M., Shokrollahi, A., Watson, M., Stockhammer, T., and L. Minder, "RaptorQ Forward Error Correction Scheme for Object Delivery", RFC 6330, August 2011.

[RFC6363] Watson, M., Begen, A., and V. Roca, "Forward Error Correction (FEC) Framework", RFC 6363, October 2011.

[RFC6364] Begen, A., "Session Description Protocol Elements for the Forward Error Correction (FEC) Framework", RFC 6364, October 2011.

[RFC6681] Watson, M., Stockhammer, T., and M. Luby, "Raptor Forward Error Correction (FEC) Schemes for FECFRAME", RFC 6681, August 2012.

[RFC6682] Watson, M., Stockhammer, T., and M. Luby, "RTP Payload Format for Raptor Forward Error Correction (FEC)", RFC 6682, August 2012.

Authors' Addresses

Giridhar Mandyam
Qualcomm Innovation Center
5775 Morehouse Drive
San Diego, California 92121
USA

Phone: +1 858 651 7200
Email: mandyam@quicinc.com

Mike Luby
Qualcomm Technologies Inc.
2030 Addison Street
Berkeley, California 94704
USA

Phone: +1 510 725 3502
Email: luby@qti.qualcomm.com

Thomas Stockhammer
Qualcomm CDMA Technologies GMBH
Franziskanerstrasse 14
Munich 81669
Germany

Phone: +49 86190959757
Email: tsto@qti.qualcomm.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: February 5, 2016

S. Nandakumar
C. Jennings
Cisco

August 04, 2015

SDP for the WebRTC
draft-nandakumar-rtcweb-sdp-08

Abstract

The Web Real-Time Communication [WebRTC] working group is charged to provide protocol support for direct interactive rich communication using audio, video and data between two peers' web browsers. With in the WebRTC framework, Session Description protocol (SDP) [RFC4566] is used for negotiating session capabilities between the peers. Such a negotiation happens based on the SDP Offer/Answer exchange mechanism described in [RFC3264].

This document provides an informational reference in describing the role of SDP and the Offer/Answer exchange mechanism for the most common WebRTC use-cases.

This SDP examples provided in this document is still a work in progress, but it aims to align closest to the evolving standards work.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 5, 2016.

Table of Contents

1. Introduction

Javascript Session Exchange Protocol(JSEP) [I-D.ietf-rtcweb-jsep] specifies a generic protocol needed to generate [RFC3264] Offers and Answers negotiated between the WebRTC peers for setting up, updating and tearing down a WebRTC session. For this purpose, SDP is used to construct [RFC3264] Offers/Answers for describing (media and non-media) streams as appropriate for the recipients of the session description to participate in the session.

The remainder of this document is organized as follows: Sections 3 and 4 provides an overview of SDP and the Offer/Answer exchange mechanism. Section 5 provides sample SDP generated for the most common WebRTC use-cases.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. SDP and the WebRTC

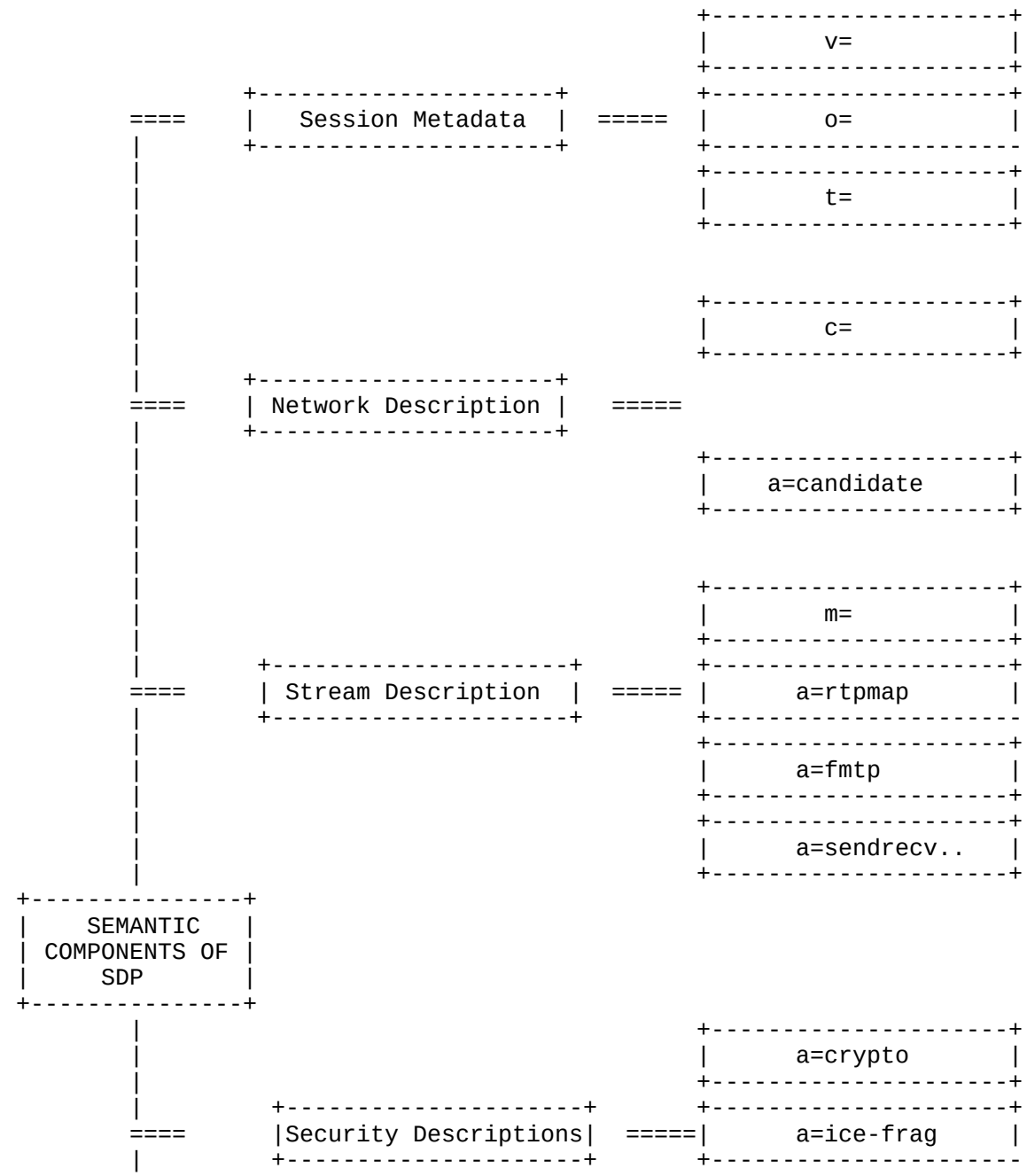
The purpose of this section is to provide a general overview of SDP and its components. For a more in-depth understanding, the readers are advised to refer to [RFC4566].

The Session Description Protocol (SDP) [RFC4566] describes multimedia sessions, which can contain audio, video, whiteboard, fax, modem, and other streams. SDP provides a general purpose, standard representation to describe various aspects of multimedia session such as media capabilities, transport addresses and related metadata in a transport agnostic manner, for the purposes of session announcement, session invitation and parameter negotiation.

As of today SDP is widely used in the context of Session Initiation Protocol [RFC3261], Real-time Transport Protocol [RFC3550] and Real-time Streaming Protocol applications [RFC2326].

Below figure introduces high-level breakup of SDP into components that semantically describe a multimedia session, in our case, a

WebRTC session [WebRTC]. It by no means captures everything about SDP and hence, should be used for informational purposes only.



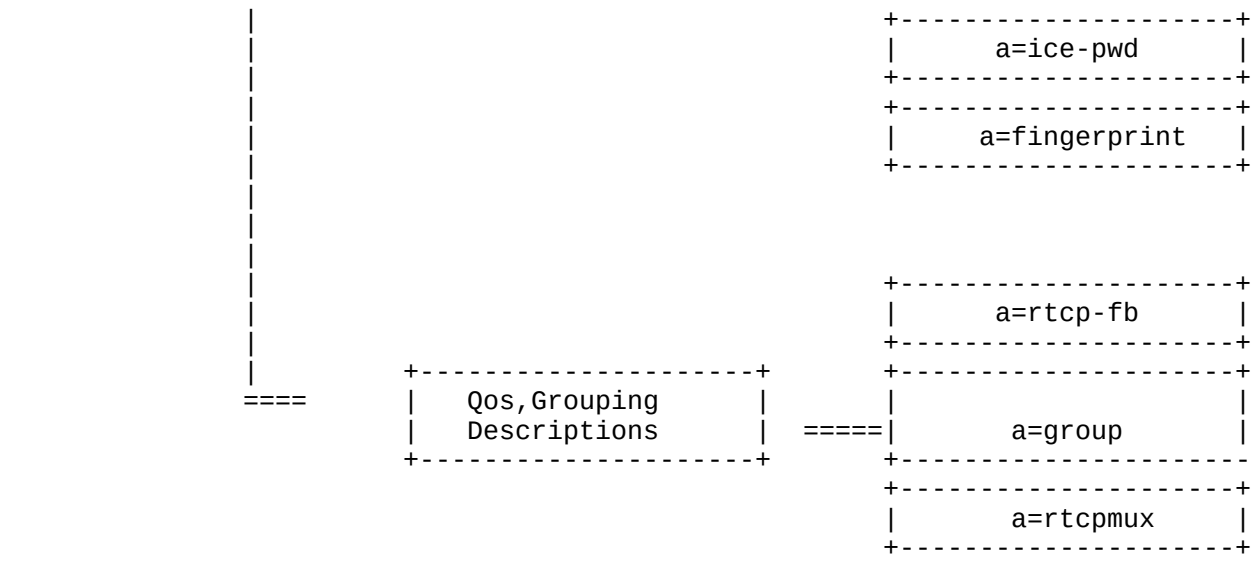


Figure 1: Semantic Components of SDP

[WebRTC] proposes JavaScript application to fully specify and control the signaling plane of a multimedia session as described in the JSEP specification [I-D.ietf-rtcweb-jsep]. JSEP provides mechanisms to create session characterization and media definition information to conduct the session based on SDP exchanges.

In this context, SDP serves two purposes:

1. Provide grammatical structure syntactically.

2. Semantically convey partipant’s intention and capabilities required to successfully negotiate a session.

4. Offer/Answer and the WebRTC

This section introduces SDP Offer/Answer Exchange mechanism mandated by WebRTC for negotiating session capabilities while setting up, updating and tearing down a WebRTC session. This section is intentionally brief in nature and interested readers are recommended to refer [RFC3264] for specific details on the protocol operation.

The Offer/Answer [RFC3264] model specifies rule for the bilateral exchange of Session Description Protocol (SDP) messages for creation of multimedia streams. It defines protocol with involved participants exchanging desired session characteristics from each others perspective constructed as SDP to negotiate the session between them.

In the most basic form,the protocol operation begins by one of the participants sending an initial SDP Offer describing its intent to start a multimedia communication session. The participant receiving the offer MAY generate an SDP Answer accepting the offer or it MAY reject the offer. If the session is accepted the Offer/Answer model guarantees a common view of the multimedia session between the participants.

At any time, either participant MAY generate a new SDP offer that updates the session in progress.

With in the context of WebRTC, the Offer/Answer model defines the state-machinery for WebRTC peers to negotiate session descriptions between them during the initial setup stages as well as for eventual session updates. Javascript Session Establishment Protocol specification [I-D.ietf-rtcweb-jsep] for WebRTC provides the mechanism for generating [RFC3264] SDP Offers and Answers in order for both sides of the session to agree upon details such as list of media formats to be sent/received, bandwidth information, crypto parameters, transport parameters, for example.

5. WebRTC Session Description Examples

A typical web based real-time multimedia communication session can be characterized as below:

- It has zero or more Audio only, Video only or Audio/Video RTP Sessions,
- MAY contain zero or more non-media data sessions,
- All the sessions are secured with DTLS-SRTP,
- Supports NAT traversal using ICE mechanism,
- Provides RTCP based feedback mechanisms,
- Sessions can be over IPv4-only, IPv6-only, dual-stack based clients.

5.1. Some Conventions

The examples given in this document follow the conventions listed below:

- In all the examples, Alice and Bob are assumed to be the WebRTC peers.
- [I-D.ietf-mmusic-sdp-bundle-negotiation] support for multiplexing several media streams over a single underlying transport is assumed by default unless explicitly specified otherwise.
- Call-flow diagrams that accompany the use-cases capture only the prominent aspects of the system behavior and intentionally is not detailed to improve readability.
- Eventhough the call-flow diagrams shows SDP being exchanged between the parties, it doesn’t represent the only way an WebRTC setup is expected to work. Other approaches may involve WebRTC applications to exchange the media setup information via non-SDP mechanisms as long as they confirm to the [I-D.ietf-rtcweb-jsep] API specification.
- The SDP examples deviate from actual on-the-wire SDP notation in several ways. This is done to facilitate readability and to conform to the restrictions imposed by the RFC formatting rules.
  - Any SDP line that is indented (compared to the initial line in the SDP block) is a continuation of the preceding line. The line break and indent are to be interpreted as a single space character.
  - Empty lines in any SDP example are inserted to make functional divisions in the SDP clearer, and are not actually part of the SDP syntax.
  - Excepting the above two conventions, line endings are to be interpreted as <CR><LF> pairs (that is, an ASCII 13 followed by an ASCII 10).
- Against each SDP line, pointers to the appropriate RFCs are provided for further informational reference. Also an attempt has been made to provide explanatory notes to enable better understanding of the SDP usage, wherever appropriate.
- Following SDP details are common across all the use-cases defined in this document unless mentioned otherwise.

* DTLS fingerprint for SRTP (a=fingerprint)

* RTP/RTCP Multiplexing (a=rtcp-mux)

* RTCP Feedback support (a=rtcp-fb)

* Host and server-reflexive candidate lines (a=candidate)

* SRTP Setup framework parameters (a=setup)

* RTCP attribute (a=rtcp)

* RTP header extension indicating audio-levels from client to the mixer

For more details, readers are recommended to refer to [I-D.ietf-rtcweb-jsep] specification.

- The term "Session" is used rather loosely in this document to refer to either a "Communication Session" or a "RTP Session" or a "RTP Stream" depending on the context.

- Payload type 109 is usually used for OPUS, 0 for PCMU, 8 for PCMA, 99 for H.264 and 120 for VP8 in most of the examples to maintain uniformity.

- In the actual use the values that represent SSRCs, ICE candidate foundations, WebRTC Mediastream and MediaStreamTrack Ids shall be much larger and random than the ones shown in the examples.

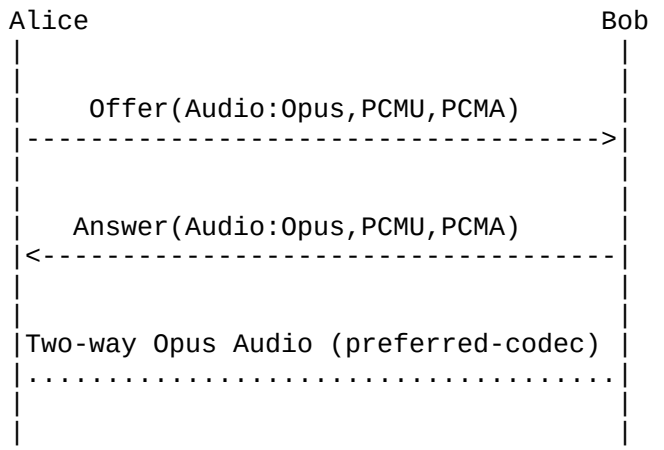
  [OPEN ISSUE-1]: SDP Examples for Data Channel, Simulcast, SVC are still being discussed and doesn’t represent the final solution.

5.2. Basic Examples

5.2.1. Audio Only Session

This common scenario shows SDP for secure two-way audio session with Alice offering Opus, PCMU, PCMA and Bob accepting all the offered audio codecs.

2-Way Audio Only Session



SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20518 0 IN IP4 0.0.0.0	[RFC4566] - Session Origin Information
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio	[I-D.ietf-mmusic-sdp-bundle-negotiation]
m=audio 54609 UDP/TLS/RTP/SAVPF 109 0 8	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=mid:audio	[RFC5888]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp-mux	[RFC5761] - Alice can perform RTP/RTCP Muxing
a=rtcp:54609 IN IP4 24.23.204.141	[RFC3605] - Port for RTCP data
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus] - Opus Codec 48khz, 2 channels
a=ptime:60	[I-D.ietf-payload-rtp-opus] - Opus packetization of 60ms
a=rtpmap:0 PCMU/8000	[RFC3551] PCMU Audio Codec
a=rtpmap:8 PCMA/8000	[RFC3551] PCMA Audio Codec
a=extmap:1 urn:ietf:params:rtp-	[RFC6464] Alice supports RTP

hdrext:ssrc-audio-level	header extension to indicate audio levels
a=sendrecv	[RFC3264] - Alice can send and recv audio
a=setup:actpass	[RFC4145] - Alice can perform DTLS before Answer arrives
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245] - DTLS Fingerprint for SRTP
a=ice-ufrag:074c6550	[RFC5245] - ICE user fragment
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245] - ICE password
a=candidate:0 1 UDP 2122194687 192.168.1.4 54609 typ host	[RFC5245] - RTP Host Candidate
a=candidate:0 2 UDP 2122194687 192.168.1.4 54609 typ host	[RFC5245] - RTCP Host Candidate
a=candidate:1 1 UDP 1685987071 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245] - RTP Server Reflexive ICE Candidate
a=candidate:1 2 UDP 1685987071 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245] - RTCP Server Reflexive Candidate
a=rtcp-fb:109 nack	[RFC5104] - Indicates NACK RTCP feedback support
a=ssrc:12345 cname:EocUG1f0fcg/yvY7	[RFC5576]
a=rtcp-rsize	[RFC5506] - Alice intends to use reduced size RTCP for this session
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]

Table 1: 5.2.1 SDP Offer

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 16833 0 IN IP4 0.0.0.0	[RFC4566] - Session Origin Information
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio	[I-D.ietf-mmusic-sdp-bundle-negotiation]
m=audio 49203 UDP/TLS/RTP/SAVPF 109 0 8	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]

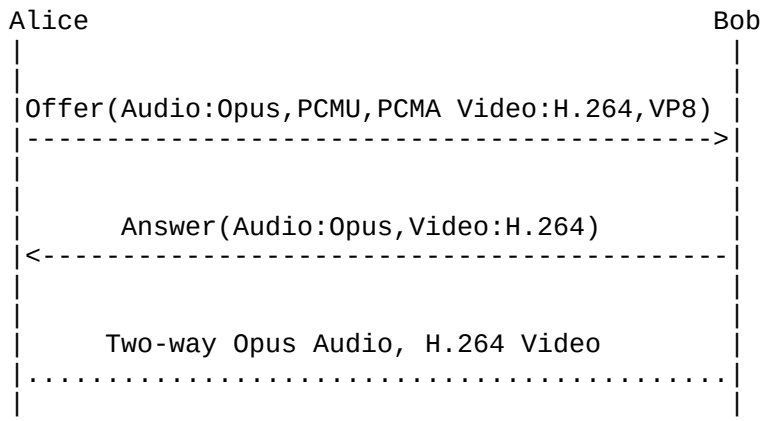
a=mid:audio	[RFC5888]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus] Opus Codec
a=ptime:60	[I-D.ietf-payload-rtp-opus] Packetization of 60ms
a=rtpmap:0 PCMU/8000	[RFC3551] PCMU Audio Codec
a=rtpmap:8 PCMA/8000	[RFC3551] PCMA Audio Codec
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464] Bob supports audio level RTP header extension as well
a=sendrecv	[RFC3264] - Bob can send and recv audio
a=setup:active	[RFC4145] - Bob carries out DTLS Handshake in parallel
a=rtcp-mux	[RFC5761] - Bob can perform RTP/RTCP Muxing on port 49203
a=fingerprint:sha-1 c9:c7:70:9d:1f:66:79:a8:07:99:41:49:83:4a:97:0e:1f:ef:6d:f7	[RFC5245] - DTLS Fingerprint for SRTP
a=ice-ufrag:05067423	[RFC5245] - ICE user fragment
a=ice-pwd:1747d1ee3474a28a397a4c3f3af08a068	[RFC5245] - ICE password parameter
a=candidate:0 1 UDP 2122194687 192.168.1.7 49203 typ host	[RFC5245] - RTP/RTCP Host ICE Candidate
a=candidate:1 1 UDP 1685987071 98.248.92.77 60654 typ srflx raddr 192.168.1.7 rport 49203	[RFC5245] - RTP/RTCP Server Reflexive ICE Candidate
a=rtcp-fb:109 nack	[RFC5104] - Indicates NACK RTCP feedback support
a=ssrc:54321 cname:NWs1ao1HmN4Xa5/yvY7	[RFC5576]
a=rtcp-rsize	[RFC5506] - Bob intends to use reduced size RTCP for this session
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]

Table 2: 5.2.1 SDP Answer

5.2.2. Audio/Video Session

Alice and Bob establish a two-way audio and video session with Opus as the audio codec and H.264 as the video codec.

2-Way Audio,Video Session



SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20518 0 IN IP4 0.0.0.0	[RFC4566] - Session Origin Information
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio video	[I-D.ietf-mmusic-sdp-bundle-negotiation]
m=audio 54609 UDP/TLS/RTP/SAVPF 109 0 8	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=mid:audio	[RFC5888]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp-mux	[RFC5761] - Alice can perform RTP/RTCP Muxing
a=rtcp:54609 IN IP4 24.23.204.141	[RFC3605] - Port for RTCP data
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus] - Opus Codec 48khz, 2 channels
a=ptime:60	[I-D.ietf-payload-rtp-opus] - Opus packetization of 60ms
a=rtpmap:0 PCMU/8000	[RFC3551] PCMU Audio Codec
a=rtpmap:8 PCMA/8000	[RFC3551] PCMA Audio Codec
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]

a=sendrecv	[RFC3264] - Alice can send and recv audio
a=setup:actpass	[RFC4145] - Alice can perform DTLS before Answer arrives
a=ice-ufrag:074c6550	[RFC5245] - ICE user fragment
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245] - ICE password parameter
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245] - DTLS Fingerprint for SRTP
a=candidate:0 1 UDP 2122194687 192.168.1.4 54609 typ host	[RFC5245] - RTP Host Candidate
a=candidate:0 2 UDP 2122194687 192.168.1.4 54609 typ host	[RFC5245] - RTCP Host Candidate
a=candidate:1 1 UDP 1685987071 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245] - RTP Server Reflexive ICE Candidate
a=candidate:1 2 UDP 1685987071 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245] - RTCP Server Reflexive Candidate.
a=rtcp-fb:109 nack	[RFC5104] - Indicates NACK RTCP feedback support
a=ssrc:12345 cname:EocUG1f0fcg/yvY7	[RFC5576]
a=rtcp-rsize	[RFC5506] - Alice intends to use reduced size RTCP for this session
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=video 54609 UDP/TLS/RTP/SAVPF 99 120	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=mid:video	[RFC5888]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtcp-mux	[RFC5761] - Alice can perform RTP/RTCP Muxing
a=rtcp:54609 IN IP4 24.23.204.141	[RFC3605] - Port for RTCP data
a=rtpmap:99 H264/90000	[RFC3984] - H.264 Video Codec
a=fmtp:99 profile-level-id=4d0028;packetization-mode=1	[RFC3984]
a=rtpmap:120 VP8/90000	[I-D.ietf-payload-vp8] - VP8 video codec
a=sendrecv	[RFC3264] - Alice can send and recv video
a=setup:actpass	[RFC4145] - Alice can perform DTLS before Answer arrives

a=ice-ufrag:074c6550	[RFC5245] - ICE user fragment
a=ice-pwd:a28a397a4c3f31747d1ee3 474af08a068	[RFC5245] - ICE password parameter
a=fingerprint:sha-1 99:41:49:83: 4a:97:0e:1f:ef:6d:f7:c9:c7:70: 9d:1f:66:79:a8:07	[RFC5245] - DTLS Fingerprint for SRTP
a=candidate:0 1 UDP 2122194687 192.168.1.4 54609 typ host	[RFC5245] - RTP Host ICE Candidate
a=candidate:0 2 UDP 2122194687 192.168.1.4 54609 typ host	[RFC5245] - RTCP Host Candidate
a=candidate:1 1 UDP 1685987071 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245] - RTP Server Reflexive ICE Candidate
a=candidate:1 2 UDP 1685987071 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245] - RTCP Server Reflexive Candidate
a=rtcp-fb:99 nack	[RFC5104] - Indicates NACK RTCP feedback support
a=rtcp-fb:99 nack pli	[RFC5104] - Indicates support for Picture loss Indication and NACK
a=rtcp-fb:99 ccm fir	[RFC5104] - Full Intra Frame Request-Codec Control Message support
a=rtcp-fb:120 nack	[RFC5104] - Indicates NACK RTCP feedback support
a=rtcp-fb:120 nack pli	[RFC5104] - Indicates support for Picture loss Indication and NACK
a=rtcp-fb:120 ccm fir	[RFC5104] - Full Intra Frame Request-Codec Control Message support
a=ssrc:1366781083 cname:EocUG1f0fcg/yvY7	[RFC5576]
a=rtcp-rsize	[RFC5506] - Alice intends to use reduced size RTCP for this session
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]

Table 3: 5.2.2 SDP Offer

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 16833 0 IN IP4 0.0.0.0	[RFC4566] - Session Origin Information

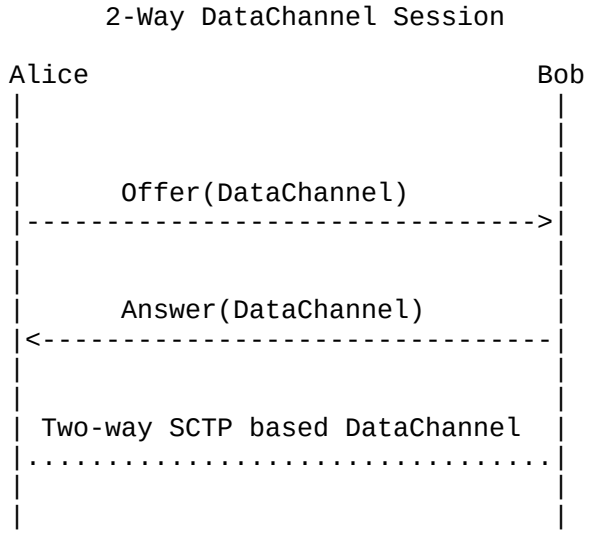
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio video	[I-D.ietf-mmusic-sdp-bundle-negotiation]
m=audio 49203 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=mid:audio	[RFC5888]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp-mux	[RFC5761] - Bob can perform RTP/RTCP Muxing
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus] - Bob accepts only Opus Codec
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:60	[I-D.ietf-payload-rtp-opus]
a=sendrecv	[RFC3264] - Bob can send and recv audio
a=setup:active	[RFC4145] - Bob carries out DTLS Handshake in parallel
a=ice-ufrag:c300d85b	[RFC5245] - ICE username frag
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245] - ICE password
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245] - DTLS Fingerprint for SRTP
a=candidate:0 1 UDP 3618095783 192.168.1.7 49203 typ host	[RFC5245] - RTP/RTCP Host ICE Candidate
a=candidate:1 1 UDP 565689203 98.248.92.77 60065 typ srflx raddr 192.168.1.7 rport 49203	[RFC5245] - RTP/RTCP Server Reflexive ICE Candidate
a=ssrc:1366788312 cname:1f0fcgEocUG/yvY7	[RFC5576]
a=rtcp-rsize	[RFC5506] - Bob intends to use reduced size RTCP for this session
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=video 49203 UDP/TLS/RTP/SAVPF 99	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=mid:video	[RFC5888]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtpmap:99 H264/90000	[RFC3984] - Bob accepts H.264

	Video Codec.
a=fmtp:99 profile-level-id=4d0028;packetization-mode=1	[RFC3984]
a=rtcp-mux	[RFC5761] - Bob can perform RTP/RTCP Muxing
a=sendrecv	[RFC3264] - Bob can send and recv video
a=setup:active	[RFC4145] - Bob carries out DTLS Handshake in parallel
a=ice-ufrag:c300d85b	[RFC5245] - ICE username frag
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245] - ICE password
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245] - DTLS Fingerprint for SRTP
a=candidate:0 1 UDP 2113667327 192.168.1.7 49203 typ host	[RFC5245] - Host ICE Candidate for Opus Stream
a=candidate:1 1 UDP 1694302207 98.248.92.77 60065 typ srflx raddr 192.168.1.7 rport 49203	[RFC5245] - Server Reflexive ICE Candidate for the above host candidate
[RFC5245] - Server Reflexive Candidate for the Second Host Candidate	a=rtcp-fb:99 nack
[RFC5104] - Indicates support for NACK based RTCP feedback	a=rtcp-fb:99 nack pli
[RFC5104] - Indicates support for Picture loss Indication and NACK	a=rtcp-fb:99 ccm fir
[RFC5104] - Full Intra Frame Request- Codec Control Message support	a=ssrc:3229706345 cname:Q/NWs1ao1HmN4Xa5
[RFC5576]	a=rtcp-rsize
[RFC5506] - Bob intends to use reduced size RTCP for this session	a=ice-options:trickle
[I-D.ietf-mmusic-trickle-ice]	

Table 4: 5.2.2 SDP Answer

5.2.3. Data Only Session

This scenario illustrates SDP negotiated to setup a data-only session based on SCTP Data Channel, thus enabling use-cases such as file-transfer for example.


2-Way DataChannel Session
Alice
Bob
Offer(DataChannel)
Answer(DataChannel)
Two-way SCTP based DataChannel

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20518 0 IN IP4 0.0.0.0	[RFC4566] - Session Origin Information
s=-	[RFC4566]
t=0 0	[RFC4566]
a=group:BUNDLE data	[I-D.ietf-mmusic-sdp-bundle-negotiation]
a=ice-ufrag:074c6550	[RFC5245] - Session Level ICE parameter
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245] - Session Level ICE parameter
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245] - Session DTLS Fingerprint for SRTP
m=application 56966 DTLS/SCTP 5000	[I-D.ietf-rtcweb-data-channel]
c=IN IP4 24.23.204.141	[RFC4566]
a=mid:data	[RFC5888]
a=sctpmap:5000 webrtc-DataChannel	[I-D.ietf-mmusic-sctp-sdp]
streams=16;label="channel 1";	
subprotocol="chat";	
a=setup:actpass	[RFC4145] - Alice can perform DTLS before Answer arrives
a=sendrecv	[RFC3264] - Alice can send and recv non-media data
a=candidate:0 1 UDP 2113667327 192.168.1.7 56966 typ host	[RFC5245]
a=candidate:1 1 UDP 1694302207 24.23.204.141 56966 typ srflx raddr 192.168.1.7 rport 56966	[RFC5245]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]

Table 5: 5.2.3 SDP Offer

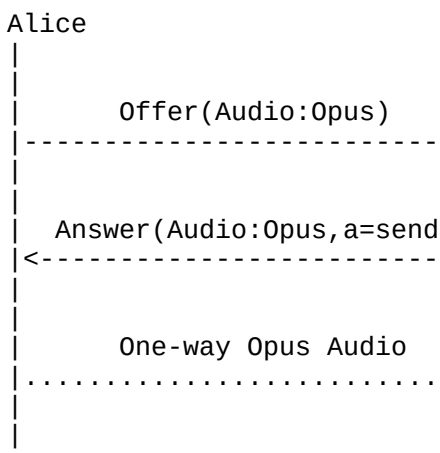
SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 16833 0 IN IP4 0.0.0.0	[RFC4566] - Session Origin Information
s=-	[RFC4566]
t=0 0	[RFC4566]
a=group:BUNDLE data	[I-D.ietf-mmusic-sdp-bundle-negotiation]
m=application 55700 DTLS/SCTP 5000	[I-D.ietf-mmusic-sctp-sdp]
c=IN IP4 98.248.92.771	[RFC4566]
a=mid:data	[RFC5888]
a=sctpmap:5000 webrtc-DataChannel:5000	[I-D.ietf-mmusic-sctp-sdp]
streams=1;label="channel 1" ;subprotocol="chat";	
a=setup:active	[RFC4145] - Bob carries out DTLS Handshake in parallel
a=sendrecv	[RFC3264] - Bob can send and recv non-media data
a=ice-ufrag:c300d85b	[RFC5245] - Session Level ICE username frag
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245] - Session Level ICE password
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245] - Session DTLS Fingerprint for SRTP
a=candidate:0 1 UDP 2113667327 192.168.1.7 55700 typ host	[RFC5245]
a=candidate:1 1 UDP 1694302207 98.248.92.77 55700 typ srflx raddr 192.168.1.7 rport 55700	[RFC5245]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]

Table 6: 5.2.3 SDP Answer

5.2.4. Audio Call On Hold

Alice calls Bob, but when Bob answers he places Alice on hold by setting the SDP direction attribute to a=sendonly in the Answer.

Audio On Hold



SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20518 0 IN IP4 0.0.0.0	[RFC4566] - Session Origin Information
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio	[I-D.ietf-mmusic-sdp-bundle-negotiation]
m=audio 54609 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=mid:audio	[RFC5888]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp-mux	[RFC5761] - Alice can perform RTP/RTCP Muxing
a=rtcp:54609 IN IP4 24.23.204.141	[RFC3605] - Port for RTCP data
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus] - Opus Codec 48khz, 2 channels
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus] - Opus packetization of 20ms
a=sendrecv	[RFC3264] - Alice can send and recv audio

a=setup:actpass	[RFC4145] - Alice can perform DTLS before Answer arrives
a=ice-ufrag:074c6550	[RFC5245] - ICE user fragment
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245] - ICE password
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245] - DTLS Fingerprint for SRTP
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:0 2 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 1685987071 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:1 2 UDP 1685987071 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=rtcp-fb:109 nack	[RFC5104] - Indicates NACK RTCP feedback support
a=ssrc:3229706345 cname:Q/NWs1ao1HmN4Xa5	[RFC5576]
a=rtcp-rsize	[RFC5506]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]

Table 7: 5.2.4 SDP Offer

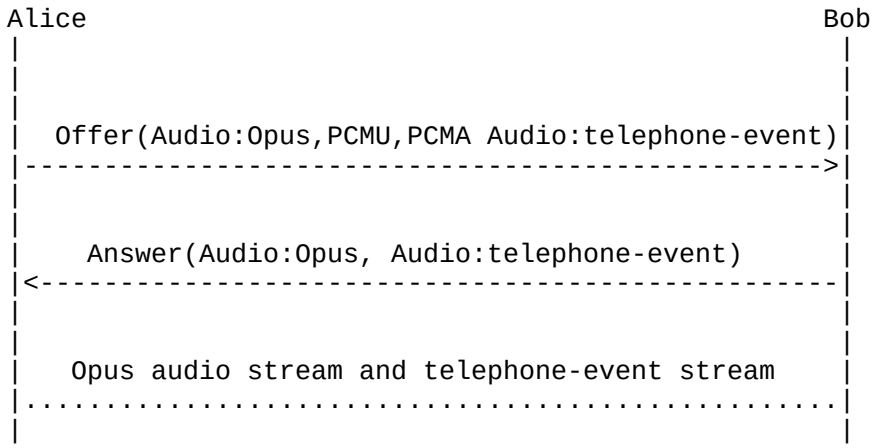
SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 16833 0 IN IP4 0.0.0.0	[RFC4566] - Session Origin Information
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio	[I-D.ietf-mmusic-sdp-bundle-negotiation]
m=audio 49203 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=mid:audio	[RFC5888]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus] - Bob accepts Opus Codec
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=sendonly	[RFC3264] - Bob puts call On Hold
a=setup:active	[RFC4145] - Bob carries out DTLS Handshake in parallel
a=rtcp-mux	[RFC5761] - Bob can perform RTP/RTCP Muxing
a=ice-ufrag:c300d85b	[RFC5245] - ICE username frag
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245] - ICE password
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d : 1f:66:79:a8:07	[RFC5245] - DTLS Fingerprint for SRTP
a=candidate:0 1 UDP 2122194687 192.168.1.7 49203 typ host	[RFC5245]
a=candidate:1 1 UDP 1685987071 98.248.92.77 49203 typ srflx raddr 192.168.1.7 rport 49203	[RFC5245]
a=ssrc:1366781083	[RFC5576]
cname:EocUG1f0fcg/yvY7	
a=rtcp-rsize	[RFC5506]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]

Table 8: 5.2.4 SDP Answer

5.2.5. Audio with DTMF Session

In this example, Alice wishes to establish two separate audio streams, one for normal audio and the other for telephone-events. Alice offers first audio stream with three codecs and the other with [RFC2833] tones (for DTMF). Bob accepts both the audio streams by choosing Opus as the audio codec and telephone-event for the other stream.

Audio Session with DTMF



SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20518 0 IN IP4 0.0.0.0	[RFC4566] - Session Origin Information
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio dtmf	[I-D.ietf-mmusic-sdp-bundle-negotiation]
m=audio 54609 UDP/TLS/RTP/SAVPF 109 0 8	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=mid:audio	[RFC5888]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp:54609 IN IP4 24.23.204.141	[RFC3605] - Port for RTCP data

a=rtcp-mux	[RFC5761] - Alice can perform RTP/RTCP Muxing
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus] - Opus Codec 48khz, 2 channels
a=ptime:20	[I-D.ietf-payload-rtp-opus] - Opus packetization of 20ms
a=rtpmap:0 PCMU/8000	[RFC3551] PCMU Audio Codec
a=rtpmap:8 PCMA/8000	[RFC3551] PCMA Audio Codec
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=sendrecv	[RFC3264] - Alice can send and recv audio
a=setup:actpass	[RFC4145] - Alice can perform DTLS before Answer arrives
a=ice-ufrag:074c6550	[RFC5245] - ICE user fragment
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245] - ICE password parameter
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245] - DTLS Fingerprint for SRTP
a=candidate:0 1 UDP 2122194687 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:0 2 UDP 2122194687 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 1685987071 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:1 2 UDP 1685987071 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=rtcp-fb:109 nack	[RFC5104] - Indicates NACK RTCP feedback support
a=ssrc:3229706345 cname:Q/NWs1ao1HmN4Xa5	[RFC5576]
a=rtcp-rsize	[RFC5506]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 54609 UDP/TLS/RTP/SAVPF 126	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=mid:dtmf	[RFC5888]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtcp-mux	[RFC5761]
a=rtcp:54609 IN IP4 24.23.204.141	[RFC3605] - Port for RTCP data
a=rtpmap:126 telephone-event/8000	[RFC2833]

a=sendonly	[RFC3264] - Alice can send DTMF Events
a=setup:actpass	[RFC4145] - Alice can perform DTLS before Answer arrives
a=ice-ufrag:074c6550	[RFC5245] - ICE user fragment
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245] - ICE password parameter
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245] - DTLS Fingerprint for SRTP
a=candidate:0 1 UDP 2122194687 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:0 2 UDP 2122194687 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 1685987071 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:1 2 UDP 1685987071 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=rtcp-fb:109 nack	[RFC5104] - Indicates NACK RTCP feedback support
a=ssrc:9032206345 cname:L/N9lk1ao1HmN4Xa5	[RFC5576]
a=rtcp-rsize	[RFC5506]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]

Table 9: 5.2.5 SDP Offer

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 16833 0 IN IP4 0.0.0.0	[RFC4566] - Session Origin Information
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio dtmf	[I-D.ietf-mmusic-sdp-bundle-negotiation]
m=audio 49203 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=mid:audio	[RFC5888]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)

a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus] - Bob accepts Opus Codec
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=sendrecv	[RFC3264] - Bob can send and receive Opus audio
a=setup:active	[RFC4145] - Bob carries out DTLS Handshake in parallel
a=rtcp-mux	[RFC5761] - Bob can perform RTP/RTCP Muxing on port 49203
a=ice-ufrag:c300d85b	[RFC5245] - ICE username frag
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245] - ICE password
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245] - Fingerprint for SRTP
a=candidate:0 1 UDP 2122194687 192.168.1.7 49203 typ host	[RFC5245]
a=candidate:1 1 UDP 1685987071 98.248.92.77 49203 typ srflx raddr 192.168.1.7 rport 49203	[RFC5245]
a=ssrc:0634322975 cname:Q/o1HmN4XNWs1aa5	[RFC5576]
a=rtcp-rsize	[RFC5506] - Alice intends to use reduced size RTCP for this session
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 49203 UDP/TLS/RTP/SAVPF 126	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=mid:dtmf	[RFC5888]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtpmap:126 telephone-event/8000	[RFC2833]
a=recvonly	[RFC3264] - Alice can receive DTMF events
a=setup:active	[RFC4145] - Bob carries out DTLS Handshake in parallel
a=rtcp-mux	[RFC5761] - Alice can perform RTP/RTCP Muxing on port 54690
a=ice-ufrag:c300d85b	[RFC5245] - ICE username frag
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245] - ICE password
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d	[RFC5245] - Fingerprint for SRTP

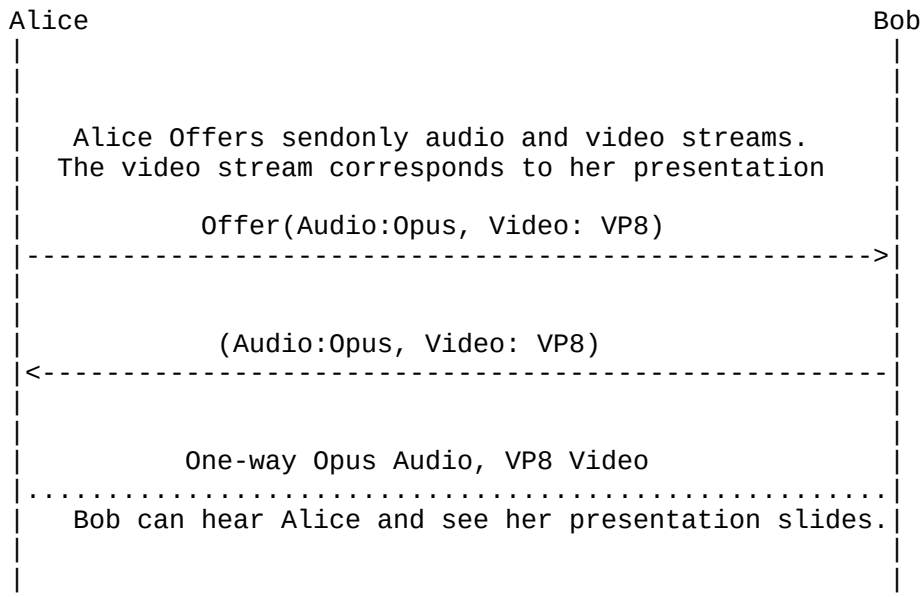
: 1f:66:79:a8:07		
a=candidate:0 1 UDP 2113667327	[RFC5245]	
192.168.1.7 49203 typ host		
a=candidate:1 1 UDP 1694302207	[RFC5245]	
98.248.92.77 49203 typ srflx		
raddr 192.168.1.7 rport 49203		
a=ssrc:6345903220	[RFC5576]	
cname:L/k1aN9lo1HmN4Xa5		
a=rtcp-rsize	[RFC5506] - Alice intends to use reduced size RTCP for this session	
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]	

Table 10: 5.2.5 SDP Answer

5.2.6. One Way Audio/Video Session - Document Camera

In this scenario Alice and Bob engage in a 1 way audio and video session with Bob receiving Alice's audio and her presentation slides as video stream.

One Way Audio & Video Session - Document Camera



SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio video	[I-D.ietf-mmusic-sdp-bundle-negotiation]
m=audio 54609 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=mid:audio	[RFC5888]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp-mux	[RFC5761]
a=rtcp:54609 IN IP4 24.23.204.141	[RFC3605] - Port for RTCP data
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=sendonly	[RFC3264] - Send only audio stream
a=setup:actpass	[RFC4145] - Alice can perform DTLS before Answer arrives
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2122194687 24.23.204.141 54609 typ host	[RFC5245]
a=candidate:0 2 UDP 2122194687 24.23.204.141 54609 typ host	[RFC5104]
a=rtcp-fb:109 nack	[RFC5104]
a=ssrc:6345903220	[RFC5576]
cname:L/k1aN9lo1HmN4Xa5	
a=rtcp-rsize	[RFC5506]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=video 54609 UDP/TLS/RTP/SAVPF 120	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=mid:video	[RFC5888]
a=msid:ma tb	Identifies RTCMediaStream ID

		(ma) and RTCMediaStreamTrack ID (tb)
a=rtcp-mux	[RFC5761]	
a=rtcp:54609 IN IP4 24.23.204.141	[RFC3605] - Port for RTCP data	
a=rtpmap:120 VP8/90000	[I-D.ietf-payload-vp8]	
a=content:slides	[RFC4796] -Alice's presentation video stream	
a=sendonly	[RFC3264] - Send only video stream	
a=setup:actpass	[RFC4145] - Alice can perform DTLS before Answer arrives	
a=ice-ufrag:074c6550	[RFC5245]	
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]	
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]	
a=candidate:0 1 UDP 2113667327 24.23.204.141 54609 typ host	[RFC5245]	
a=candidate:0 2 UDP 2113667326 24.23.204.141 54609 typ host	[RFC5104]	
a=rtcp-fb:120 nack	[RFC5104]	
a=rtcp-fb:120 nack pli	[RFC5104]	
a=rtcp-fb:120 ccm fir	[RFC5104]	
a=ssrc:3429951804 cname:Q/NWs1ao1HmN4Xa5	[RFC5576]	
a=rtcp-rsize	[RFC5506]	
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]	

Table 11: 5.2.6 SDP Offer

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 16833 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio video	[I-D.ietf-mmusic-sdp-bundle-negotiation]
m=audio 49203 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=mid:audio	[RFC5888]
a=msid:ma ta	Identifies RTCMediaStream ID

	(ma) and RTCMediaStreamTrack ID (ta)
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=recvonly	[RFC3264] - Receive only audio stream
a=setup:active	[RFC4145] - Bob carries out DTLS Handshake in parallel
a=rtcp-mux	[RFC5761]
a=ice-ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d : 1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 98.248.92.77 49203 typ host	[RFC5245]
a=ssrc:9513429804 cname:Q/o1HmNWs1aN4Xa5	[RFC5576]
a=rtcp-rsize	[RFC5506]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=video 49203 UDP/TLS/RTP/SAVPF 120	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=mid:video	[RFC5888]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtpmap:120 VP8/90000	[I-D.ietf-payload-vp8]
a=content:slides	[RFC4796]
a=recvonly	[RFC3264] - Receive Only Alice’s presentation stream
a=setup:active	[RFC4145] - Bob carries out DTLS Handshake in parallel
a=rtcp-mux	[RFC5761]
a=ice-ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 98.248.92.77 49203 typ host	[RFC5245]
a=ssrc:1366781083 cname:EocUG1f0fcg/yvY7	[RFC5576]
a=rtcp-rsize	[RFC5506]

a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]

Table 12: 5.2.6 SDP Answer

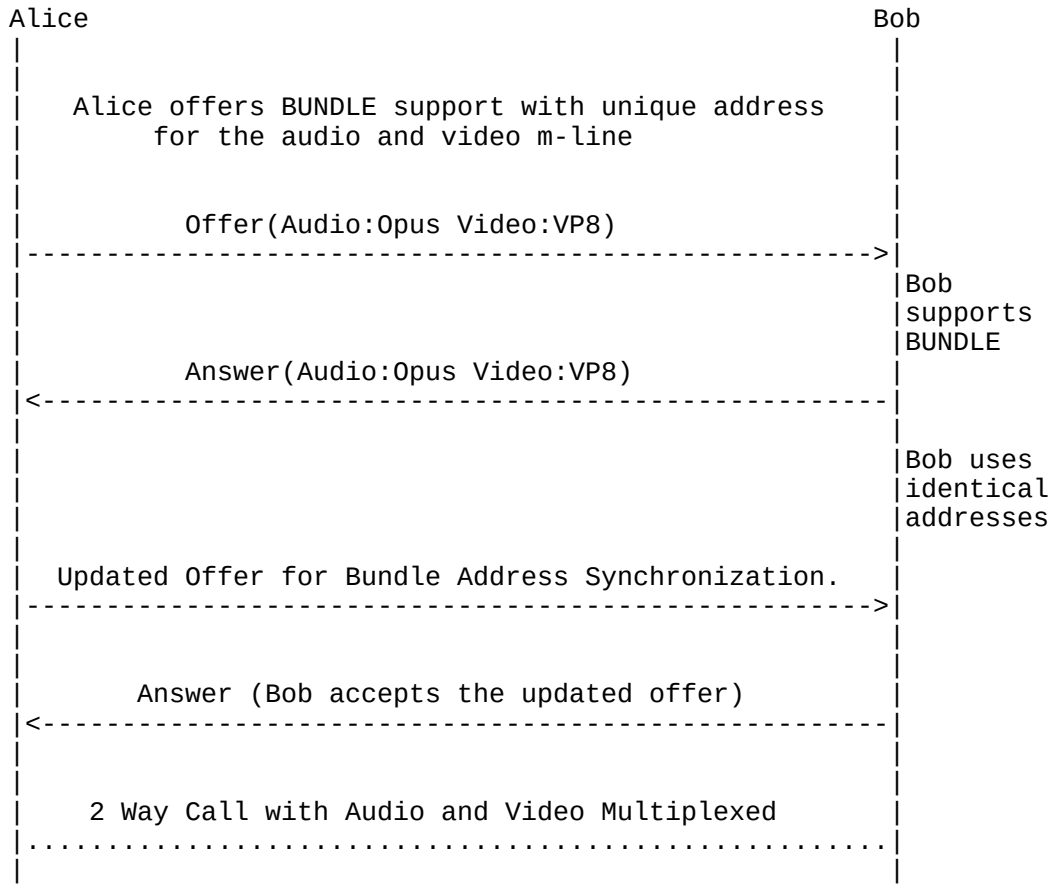
5.2.7. Audio, Video Session with BUNDLE Support Unknown

In this example, since Alice is unsure of the Bob's support of the BUNDLE framework, following 3 step procedures are performed in order to negotiate and setup a BUNDLE Address for the session

- An SDP Offer, in which the Alice assigns unique addresses to each "m=" line in the BUNDLE group, and requests the Answerer to select the Offerer's BUNDLE address.
- An SDP Answer, in which the Bob indicates its support for BUNDLE, and assigns its own BUNDLE address for the BUNDLED m= lines.
- A subsequent SDP Offer from Alice, which is used to perform BUNDLE Address Synchronization (BAS).

Once the Offer/Answer exchange completes, both Alice and Bob each end up using single RTP Session for both the Media Streams.

Two-Way Secure Audio,Video with BUNDLE support unknown



SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20518 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio video	[I-D.ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
m=audio 54609 UDP/TLS/RTP/SAVPF 109	[RFC4566]

c=IN IP4 24.23.204.141	[RFC4566]
a=mid:audio	[RFC5888] Audio m=line part of BUNDLE group with a unique port number
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp-mux	[RFC5761]
a=rtcp:54609 IN IP4 24.23.204.141	[RFC3605]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=sendrecv	[RFC3264]
a=setup:actpass	[RFC4145] - Alice can perform DTLS before Answer arrives
a=ssrc:11111	[RFC5576]
cname:EocUG1f0fcg/yvY7	
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2122194687 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:0 2 UDP 2122194687 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 1685987071 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:1 2 UDP 1685987071 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=rtcp-fb:109 nack	[RFC5104]
a=rtcp-rsize	[RFC5506]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=video 62537 UDP/TLS/RTP/SAVPF 120	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=mid:video	[RFC5888] Video m=line part of the Bundle group with a unique port number
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtcp-mux	[RFC5761]

a=rtcp:62537 IN IP4 24.23.204.141	[RFC3605] - Port for RTCP data
a=rtpmap:120 VP8/90000	[I-D.ietf-payload-vp8]
a=sendrecv	[RFC3264]
a=setup:actpass	[RFC4145] - Alice can perform DTLS before Answer arrives
a=ssrc:22222 cname:Q/NWs1ao1HmN4Xa5	[RFC5576]
a=ice-ufrag:6550074c	[RFC5245]
a=ice-pwd:74af08a068a28a397a4c3f 31747d1ee34	[RFC5245]
a=fingerprint:sha-1 1f:ef:6d:f7: c9:c7:70:9d:1f:66:99:41:49:83: 4a:97:0e79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2122194687 192.168.1.4 62537 typ host	[RFC5245]
a=candidate:0 2 2122194687 192.168.1.4 62537 typ host	[RFC5245]
a=candidate:1 1 UDP 1685987071 24.23.204.141 62537 typ srflx raddr 192.168.1.4 rport 62537	[RFC5245]
a=candidate:1 2 UDP 1685987071 24.23.204.141 62537 typ srflx raddr 192.168.1.4 rport 62537	[RFC5245]
a=rtcp-fb:120 nack	[RFC5104]
a=rtcp-fb:120 nack pli	[RFC5104]
a=rtcp-fb:120 ccm fir	[RFC5104]
a=rtcp-rsize	[RFC5506]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]

Table 13: 5.2.7 SDP Offer w/BUNDLE

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 16833 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio video	[I-D.ietf-mmusic-sdp-bundle-negotiation] Bob supports BUNDLE semantics.
m=audio 49203 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID

		(ma) and RTCMediaStreamTrack ID (ta)
a=mid:audio		[RFC5888] Audio m=line part of the BUNDLE group
a=rtpmap:109 opus/48000/2		[I-D.ietf-payload-rtp-opus]
a=ptime:20		[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level		[RFC6464]
a=sendrecv		[RFC3264]
a=setup:active		[RFC4145] - Bob carries out DTLS Handshake in parallel
a=rtcp-fb:109 nack		[RFC5104]
a=rtcp-mux		[RFC5761]
a=ssrc:33333 cname:Q/1HmN4Xa5NWs1ao		[RFC5576]
a=ice-ufrag:c300d85b		[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2		[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07		[RFC5245]
a=candidate:0 1 UDP 2122194687 192.168.1.7 49203 typ host		[RFC5245]
a=candidate:1 1 UDP 1685987071 98.248.92.77 49203 typ srflx raddr 192.168.1.7 rport 49203		[RFC5245]
a=rtcp-rsize		[RFC5506]
a=ice-options:trickle		[I-D.ietf-mmusic-trickle-ice]
m=video 49203 UDP/TLS/RTP/SAVPF 120		[RFC4566]
c=IN IP4 98.248.92.77		[RFC4566]
a=mid:video		[RFC5888] Video m=line part of the BUNDLE group with the port from audio line repeated
a=msid:ma tb		Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtpmap:120 VP8/90000		[I-D.ietf-payload-vp8]
a=sendrecv		[RFC3264]
a=setup:active		[RFC4145] - Bob carries out DTLS Handshake in parallel
a=rtcp-mux		[RFC5761]
a=ssrc:44444 cname:Q/2AqlmN4Xa5NWs		[RFC5576]
a=ice-ufrag:85bc300d		[RFC5245]
a=ice-pwd:bd2de4e9991c325921d5d47efbabd9a2		[RFC5245]
a=fingerprint:sha-1 41:49:83:4a:		[RFC5245]

99:97:0e:1f:ef:6d:f7:c9:c7:70: 9d:1f:66:79:a8:07	
a=candidate:0 1 UDP 2122194687 192.168.1.7 49203 typ host	[RFC5245]
a=candidate:1 1 UDP 1685987071 98.248.92.77 49203 typ srflx raddr 192.168.1.7 rport 49203	[RFC5245]
a=rtcp-fb:120 nack	[RFC5104]
a=rtcp-fb:120 nack pli	[RFC5104]
a=rtcp-fb:120 ccm fir	[RFC5104]
a=rtcp-rsize	[RFC5506]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]

Table 14: 5.2.7 SDP Answer w/BUNDLE

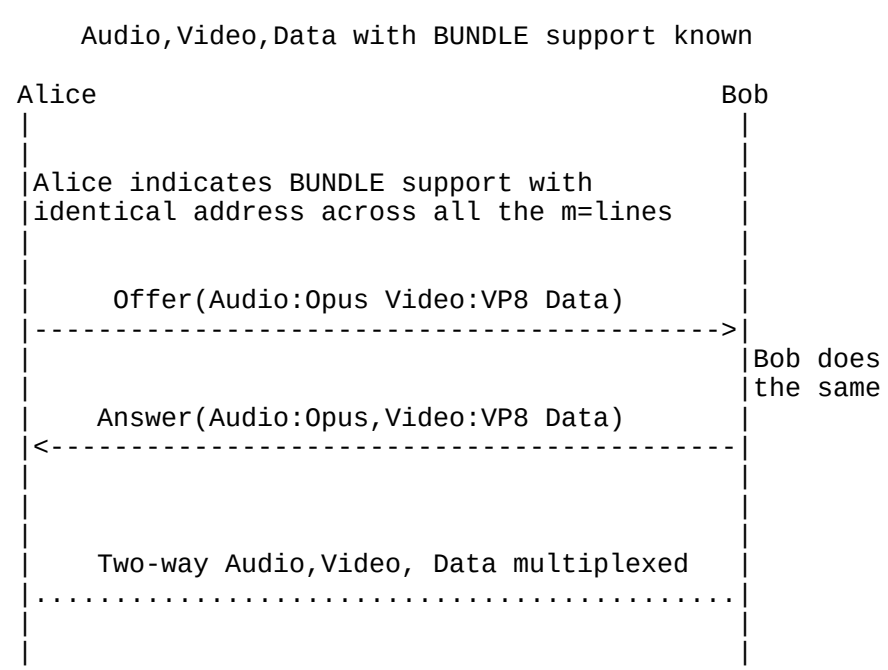
SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20518 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio video	[I-D.ietf-mmusic-sdp-bundle-negotiation]
m=audio 54609 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=mid:audio	[RFC5888] - Port number finalized as Bundle Address.
a=rtcp-mux	[RFC5761]
a=rtcp:54609 IN IP4 24.23.204.141	[RFC3605]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=sendrecv	[RFC3264]
a=setup:actpass	[RFC4145]
a=ssrc:11111 cname:EocUG1f0fcg/yvY7	[RFC5576]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]

a=fingerprint:sha-1 99:41:49:83: 4a:97:0e:1f:ef:6d:f7:c9:c7:70: 9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2122194687 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 1685987071 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=rtcp-fb:109 nack	[RFC5104]
a=rtcp-rsize	[RFC5506]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=video 54609 UDP/TLS/RTP/SAVPF 120	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=mid:video	[RFC5888]
a=rtpmap:120 VP8/90000	[I-D.ietf-payload-vp8]
a=sendrecv	[RFC3264]
a=setup:actpass	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ssrc:22222 cname:Q/NWs1ao1HmN4Xa5	[RFC5576]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3 474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83: 4a:97:0e:1f:ef:6d:f7:c9:c7:70: 9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2122194687 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 1685987071 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=rtcp-fb:120 nack	[RFC5104]
a=rtcp-fb:120 nack pli	[RFC5104]
a=rtcp-fb:120 ccm fir	[RFC5104]
a=rtcp-rsize	[RFC5506]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]

Table 15: 5.2.7 SDP Offer for BAS

5.2.8. Audio, Video and Data Session

This example shows SDP for negotiating a session with Audio, Video and data streams between Alice and Bob with BUNDLE support known.



SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20518 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio video data	[I-D.ietf-mmusic-sdp-bundle-negotiation]
m=audio 54609 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp:54609 IN IP4	[RFC3605]

24.23.204.141	
a=mid:audio	[RFC5888]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=sendrecv	[RFC3264]
a=setup:actpass	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2122194687 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:0 2 UDP 2122194687 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 1 UDP 1685987071 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:1 2 UDP 1685987071 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=rtcp-fb:109 nack	[RFC5104]
a=ssrc:11111	[RFC5576]
cname:Q/NWs1ao1HmN4Xa5	
a=rtcp-rsize	[RFC5506]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=video 54609 UDP/TLS/RTP/SAVPF 120	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtcp:54609 IN IP4 24.23.204.141	[RFC3605]
a=mid:video	[RFC5888]
a=rtpmap:120 VP8/90000	[I-D.ietf-payload-vp8]
a=sendrecv	[RFC3264]
a=setup:actpass	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:	[RFC5245]

9d:1f:66:79:a8:07	
a=candidate:0 1 UDP 2122194687 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:0 2 UDP 2122194687 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 1685987071 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:1 2 UDP 1685987071 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=rtcp-fb:120 nack	[RFC5104]
a=rtcp-fb:120 nack pli	[RFC5104]
a=rtcp-fb:120 ccm fir	[RFC5104]
a=ssrc:22222 cname:Q/aoNWs11HmN4Xa5	[RFC5576]
a=rtcp-rsize	[RFC5506]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=application 54609 DTLS/SCTP 5000	[I-D.ietf-rtcweb-data-channel]
c=IN IP4 24.23.204.141	[RFC4566]
a=mid:data	[RFC5888]
a=sctpmap:5000 webrtc-DataChannel streams=1;label="channel 1"; subprotocol="chat";	[I-D.ietf-mmusic-sctp-sdp]
a=sendrecv	[RFC3264]
a=setup:actpass	[RFC4145]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3 474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83: 4a:97:0e:1f:ef:6d:f7:c9:c7:70: 9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2122194687 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 1685987071 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]

Table 16: 5.2.8 SDP Offer

SDP Contents	RFC#/Notes
v=0	[RFC4566]

o=- 16833 0 IN IP4 0.0.0.0	[RFC4566] - Session Origin Information
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio video data	[I-D.ietf-mmusic-sdp-bundle-negotiation]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice] Bob's trickle support support is indicated at the session level
m=audio 49203 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=mid:audio	[RFC5888]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=sendrecv	[RFC3264]
a=setup:active	[RFC4145]
a=rtcp-mux	[RFC5761]
a=rtcp-fb:109 nack	[RFC5104]
a=ice-ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2122194687 192.168.1.7 49203 typ host	[RFC5245]
a=candidate:1 1 UDP 1685987071 98.248.92.77 49203 typ srflx raddr 192.168.1.7 rport 49203	[RFC5245]
a=ssrc:33333 cname:L/aoNWs11HmN4Xa5	[RFC5576]
a=rtcp-rsize	[RFC5506]
m=video 49203 UDP/TLS/RTP/SAVPF 120	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=mid:video	[RFC5888]
a=rtpmap:120 VP8/90000	[I-D.ietf-payload-vp8]

a=sendrecv	[RFC3264]
a=setup:active	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ice-ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d4 7efbabd9a2	[RFC5245]
a=fingerprint:sha-1 99:41:49:83: 4a:97:0e:1f:ef:6d:f7:c9:c7:70: 9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2122194687 192.168.1.7 49203 typ host	[RFC5245]
a=candidate:1 1 UDP 1685987071 98.248.92.77 49203 typ srflx raddr 192.168.1.7 rport 49203	[RFC5245]
a=rtcp-fb:120 nack	[RFC5104]
a=rtcp-fb:120 nack pli	[RFC5104]
a=rtcp-fb:120 ccm fir	[RFC5104]
a=ssrc:44444 cname:EocUG1f0fcg/yvY7	[RFC5576]
a=rtcp-rsize	[RFC5506]
m=application 49203 DTLS/SCTP 5000	[I-D.ietf-mmusic-sctp-sdp]
c=IN IP4 98.248.92.771	[RFC4566]
a=mid:data	[RFC5888]
a=sctpmap:5000 webrtc-DataChannel streams=16;label="channel 1"; subprotocol="chat";	[I-D.ietf-mmusic-sctp-sdp]
a=setup:active	[RFC4145]
a=sendrecv	[RFC3264]
a=ice-ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d4 7efbabd9a2	[RFC5245]
a=fingerprint:sha-1 99:41:49:83: 4a:97:0e:1f:ef:6d:f7:c9:c7:70: 9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2122194687 192.168.1.7 49203 typ host	[RFC5245]
a=candidate:1 1 UDP 1685987071 98.248.92.77 49203 typ srflx raddr 192.168.1.7 rport 49203	[RFC5245]

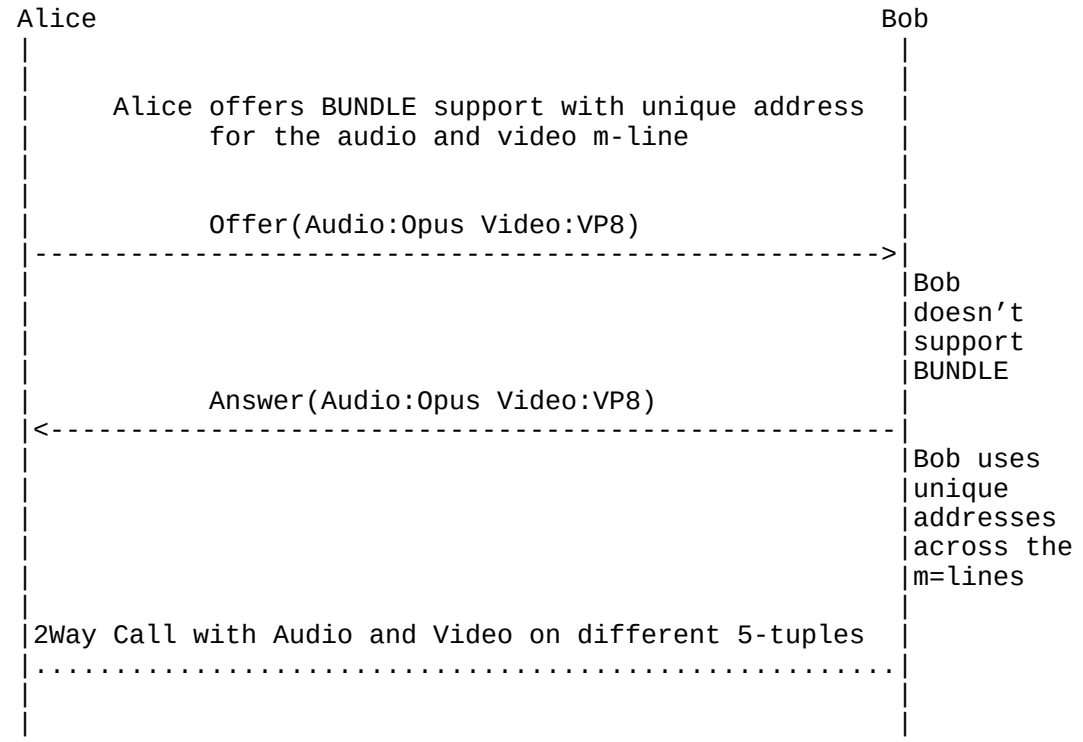
Table 17: 5.2.8 SDP Answer

5.2.9. Audio, Video Session with BUNDLE Unsupported

This use-case illustrates SDP Offer/Answer exchange where the far-end (Bob) either doesn't support media bundling or doesn't want to group m=lines over a single 5-tuple.

On successful Offer/Answer exchange, Alice and Bob each end up using unique 5-tuple for audio and video media streams respectively.

Two-Way Secure Audio,Video with BUNDLE Unsupported



SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20518 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]

a=group:BUNDLE audio video	[I-D.ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 55232 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=mid:audio	[RFC5888] Audio m=line part of BUNDLE group with a unique port number
a=rtcp:55232 IN IP4 24.23.204.141	[RFC3605]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=setup:actpass	[RFC4145] - Alice can perform DTLS before Answer arrives
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=rtcp-fb:109 nack	[RFC5104]
a=ssrc:11111 cname:EocUG1f0fcg/yvY7	[RFC5576]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d : 1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2122194687 192.168.1.4 55232 typ host	[RFC5245]
a=candidate:0 2 UDP 2122194687 192.168.1.4 55232 typ host	[RFC5245]
a=candidate:1 1 UDP 1685987071 24.23.204.141 55232 typ srflx raddr 192.168.1.4 rport 55232	[RFC5245]
a=candidate:1 2 UDP 1685987071 24.23.204.141 55232 typ srflx raddr 192.168.1.4 rport 55232	[RFC5245]
a=rtcp-rsize	[RFC5506]
m=video 54332 UDP/TLS/RTP/SAVPF 120	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID

		(ma) and RTCMediaStreamTrack ID (tb)
a=mid:video		[RFC5888] Video m=line part of the BUNDLE group with a unique port number
a=rtcp:54332 IN IP4 24.23.204.141		[RFC3605]
a=rtpmap:120 VP8/90000		[I-D.ietf-payload-vp8]
a=sendrecv		[RFC3264]
a=setup:actpass		[RFC4145] - Alice can perform DTLS before Answer arrives
a=rtcp-mux		[RFC5761]
a=ssrc:22222 cname:yvY7/EocUG1f0fcg		[RFC5576]
a=ice-ufrag:7872093		[RFC5245]
a=ice-pwd:ee3474af08a068a28a397a4c3f31747d1		[RFC5245]
a=fingerprint:sha-1 6d:f7:c9:c7:70:9d:1f:66:79:a8:07:99:41:49:83:4a:97:0e:1f:ef		[RFC5245]
a=candidate:0 1 UDP 2122194687 192.168.1.4 54332 typ host		[RFC5245]
a=candidate:0 2 2122194687 192.168.1.4 54332 typ host		[RFC5245]
a=candidate:1 1 UDP 1685987071 24.23.204.141 54332 typ srflx raddr 192.168.1.4 rport 54332		[RFC5245]
a=candidate:1 2 UDP 1685987071 24.23.204.141 54332 typ srflx raddr 192.168.1.4 rport 54332		[RFC5245]
a=rtcp-fb:120 nack		[RFC5104]
a=rtcp-fb:120 nack pli		[RFC5104]
a=rtcp-fb:120 ccm fir		[RFC5104]
a=rtcp-rsize		[RFC5506]

Table 18: 5.2.9 SDP Offer w/BUNDLE

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 16833 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]

m=audio 53214 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp:60065 IN IP4 98.248.92.77	[RFC3605]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext :ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=setup:active	[RFC4145] - Bob carries out DTLS Handshake in parallel
a=sendrecv	[RFC3264]
a=rtcp-fb:109 nack	[RFC5104]
a=ice-ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97 :0e:1f:ef:6d:f7:c9:c7:70: 9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2122194687 192.168.1.7 53214 typ host	[RFC5245]
a=candidate:1 1 UDP 1685987071 98.248.92.77 53214 typ srflx raddr 192.168.1.7 rport 53214	[RFC5245]
a=candidate:0 2 UDP 2122194687 192.168.1.7 60065 typ host	[RFC5245]
a=candidate:1 2 UDP 1685987071 98.248.92.77 60065 typ srflx raddr 192.168.1.7 rport 60065	[RFC5245]
a=rtcp-rsize	[RFC5506]
m=video 58679 UDP/TLS/RTP/SAVPF 120	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtcp:56507 IN IP4 98.248.92.77	[RFC3605]
a=rtpmap:120 VP8/90000	[I-D.ietf-payload-vp8]
a=setup:active	[RFC4145] - Bob carries out DTLS Handshake in parallel
a=sendrecv	[RFC3264]
a=ice-ufrag:85bC300	[RFC5245]

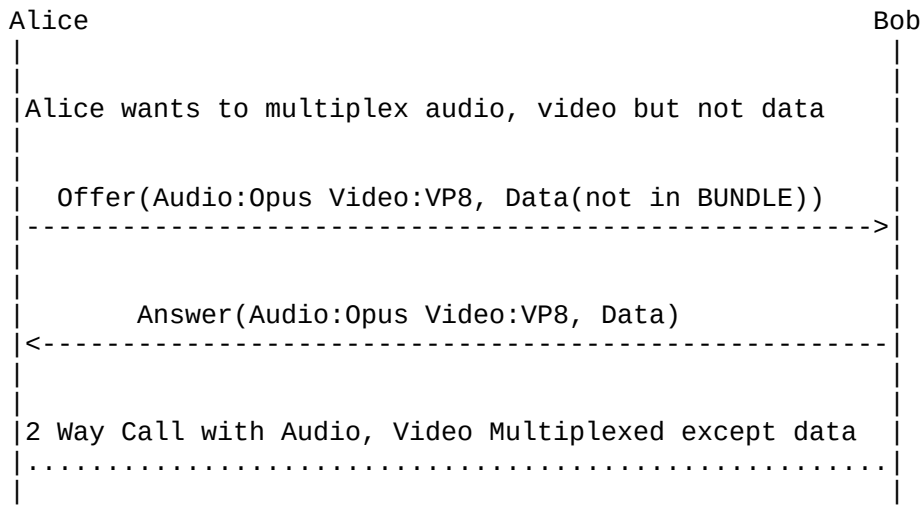
a=ice-pwd:325921d5d47efbabd9a2de4e99bd291c	[RFC5245]
a=fingerprint:sha-1 9d:1f:66:79:a8:07 :99:41:49:83:4a:97:0e:1f: ef:6d:f7:c9:c7:70	[RFC5245]
a=candidate:0 1 UDP 2122194687 192.168.1.7 58679 typ host	[RFC5245]
a=candidate:1 1 UDP 1685987071 98.248.92.77 58679 typ srflx raddr 192.168.1.7 rport 58679	[RFC5245]
a=candidate:0 1 UDP 2122194687 192.168.1.7 56507 typ host	[RFC5245]
a=candidate:1 1 UDP 1685987071 98.248.92.77 56507 typ srflx raddr 192.168.1.7 rport 58679	[RFC5245]
a=rtcp-fb:120 nack	[RFC5104]
a=rtcp-fb:120 nack pli	[RFC5104]
a=rtcp-fb:120 ccm fir	[RFC5104]
a=rtcp-rsize	[RFC5506]

Table 19: 5.2.9 SDP Answer without BUNDLE

5.2.10. Audio, Video BUNDLED, but Data (Not BUNDLED)

This example show-cases SDP for negotiating a session with Audio, Video and data streams between Alice and Bob with data stream not being part of the BUNDLE group. This is shown by assigning unique port for data media sections.

Audio, Video, with Data (Not in BUNDLE)



SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20518 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio video	[I-D.ietf-mmusic-sdp-bundle-negotiation] Alice wants to BUNDLE only audio and video media.
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 54609 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp:54609 IN IP4 24.23.204.141	[RFC3605]
a=mid:audio	[RFC5888]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]

a=sendrecv	[RFC3264]
a=setup:actpass	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3 474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83: 4a:97:0e:1f:ef:6d:f7:c9:c7:70: 9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 54609 typ host	[RFC5245]
a=rtcp-fb:109 nack	[RFC5104]
a=ssrc:11111 cname:Q/NWs1ao1HmN4Xa5	[RFC5576]
a=rtcp-rsize	[RFC5506]
m=video 54609 UDP/TLS/RTP/SAVPF 120	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtcp:54609 IN IP4 24.23.204.141	[RFC3605]
a=mid:video	[RFC5888]
a=rtpmap:120 VP8/90000	[I-D.ietf-payload-vp8]
a=sendrecv	[RFC3264]
a=setup:actpass	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3 474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83: 4a:97:0e:1f:ef:6d:f7:c9:c7:70: 9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 54609 typ host	[RFC5245]
a=rtcp-fb:120 nack	[RFC5104]
a=rtcp-fb:120 nack pli	[RFC5104]
a=rtcp-fb:120 ccm fir	[RFC5104]
a=ssrc:22222 cname:Q/aoNWs11HmN4Xa5	[RFC5576]
a=rtcp-rsize	[RFC5506]
m=application 10000 DTLS/SCTP 5000	[I-D.ietf-rtcweb-data-channel]

c=IN IP4 24.23.204.141	[RFC4566]
a=mid:data	[RFC5888]
a=sctpmap:5000 webrtc-DataChannel streams=16;label="channel 1"; subprotocol="chat";	[I-D.ietf-mmusic-sctp-sdp]
a=sendrecv	[RFC3264]
a=setup:actpass	[RFC4145]
a=ice-ufrag:89819013	[RFC5245]
a=ice-pwd:1747d1ee3474af08a068a28a397a4c3f3	[RFC5245]
a=fingerprint:sha-1 0e:1f:ef:6d:f7:c9:c7:70:99:41:49:83:4a:97:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.4 10000 typ host	[RFC5245]

Table 20: 5.2.10 SDP Offer

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 16833 0 IN IP4 0.0.0.0	[RFC4566] - Session Origin Information
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio video	[I-D.ietf-mmusic-sdp-bundle-negotiation]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 49203 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=mid:audio	[RFC5888]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=sendrecv	[RFC3264]
a=setup:active	[RFC4145]
a=rtcp-mux	[RFC5761]
a=rtcp-fb:109 nack	[RFC5104]
a=ice-ufrag:c300d85b	[RFC5245]

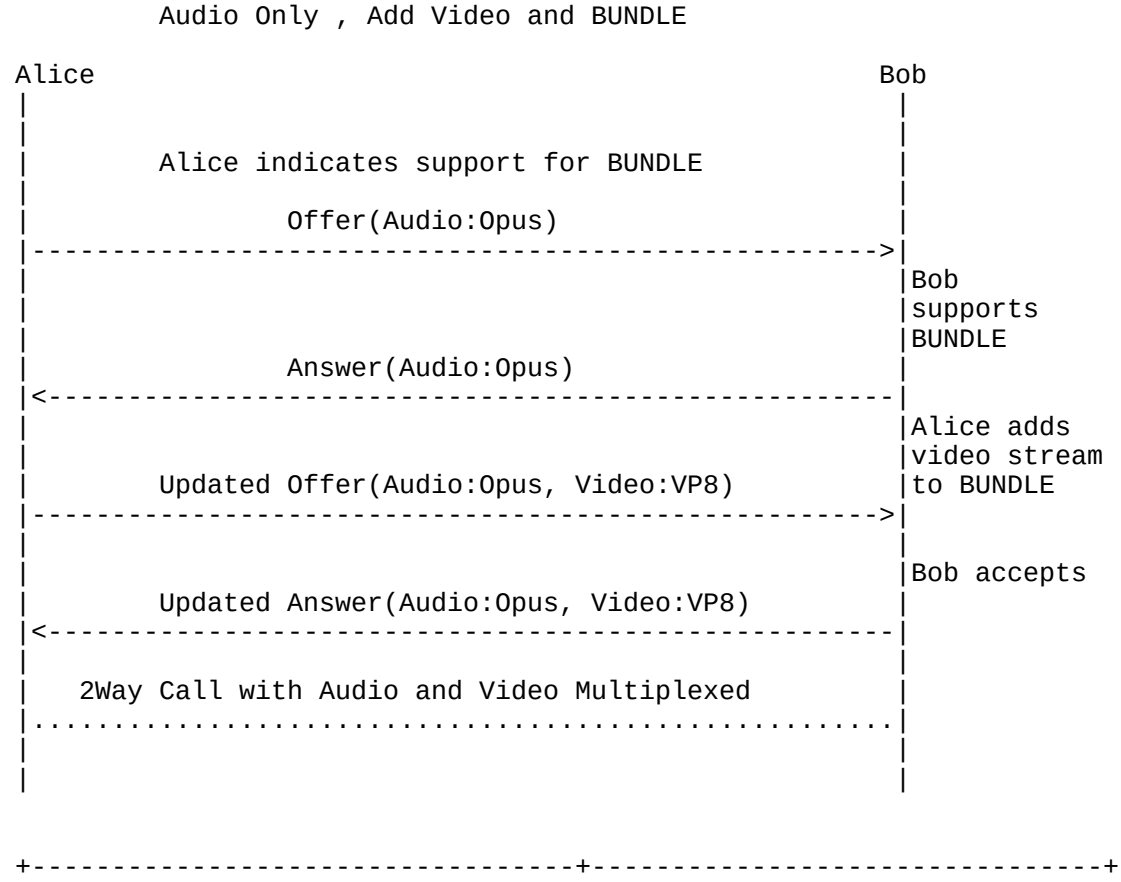
a=ice-pwd:de4e99bd291c325921d5d4 7efbabd9a2	[RFC5245]
a=fingerprint:sha-1 99:41:49:83: 4a:97:0e:1f:ef:6d:f7:c9:c7: 70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.7 49203 typ host	[RFC5245]
a=ssrc:33333 cname:L/aoNWs11HmN4Xa5	[RFC5576]
a=rtcp-rsize	[RFC5506]
m=video 49203 UDP/TLS/RTP/SAVPF 120	[RFC4566]
c=IN IP4 98.248.92.771	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=mid:video	[RFC5888]
a=rtpmap:120 VP8/90000	[I-D.ietf-payload-vp8]
a=sendrecv	[RFC3264]
a=setup:active	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ice-ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d4 7efbabd9a2	[RFC5245]
a=fingerprint:sha-1 99:41:49:83: 4a:97:0e:1f:ef:6d:f7:c9:c7:70: 9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.7 49203 typ host	[RFC5245]
a=rtcp-fb:120 nack	[RFC5104]
a=rtcp-fb:120 nack pli	[RFC5104]
a=rtcp-fb:120 ccm fir	[RFC5104]
a=ssrc:44444 cname:EocUG1f0fcg/yvY7	[RFC5576]
a=rtcp-rsize	[RFC5506]
m=application 20000 DTLS/SCTP 5000	[I-D.ietf-mmusic-sctp-sdp]
c=IN IP4 98.248.92.77	[RFC4566]
a=mid:data	[RFC5888]
a=sctpmap:5000 webrtc-DataChannel streams=1;label="channel 1"; subprotocol="chat";	[I-D.ietf-mmusic-sctp-sdp]
a=setup:active	[RFC4145]
a=sendrecv	[RFC3264]
a=ice-ufrag:991Ca2a5e	[RFC5245]
a=ice-pwd:921d5d47efbabd9a2de4e9 9bd291c325	[RFC5245]

a=fingerprint:sha-1 6d:f7:c9:c7: 70:9d:1f:66:79:a8:07:99:41:49: 83:4a:97:0e:1f:ef	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.7 20000 typ host	[RFC5245]

Table 21: 5.2.10 SDP Answer

5.2.11. Audio Only, Add Video to BUNDLE

This example involves 2 Offer/Answer exchanges. First one setting up Audio-only session followed by an updated Offer/Answer exchange to add video stream to the ongoing session. Also the newly added video stream is BUNDLED with the audio stream.



SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20518 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio	[I-D.ietf-mmusic-sdp-bundle-negotiation] Alice wants to BUNDLE only audio and video media.
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 54609 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp:64678 IN IP4 24.23.204.141	[RFC3605]
a=mid:audio	[RFC5888]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=sendrecv	[RFC3264]
a=setup:actpass	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
a=rtcp-fb:109 nack	[RFC5104]
a=ssrc:11111 cname:Q/NWs1ao1HmN4Xa5	[RFC5576]
a=rtcp-rsize	[RFC5506]

Table 22: 5.2.11 SDP Offer

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 16833 0 IN IP4 0.0.0.0	[RFC4566] - Session Origin Information
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio	[I-D.ietf-mmusic-sdp-bundle-negotiation]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 49203 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp:60065 IN IP4 98.248.92.77	[RFC3605]
a=mid:audio	[RFC5888]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=sendrecv	[RFC3264]
a=setup:active	[RFC4145]
a=rtcp-mux	[RFC5761]
a=rtcp-fb:109 nack	[RFC5104]
a=ice-ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.7 49203 typ host	[RFC5245]
a=candidate:1 1 UDP 1694302207 98.248.92.77 49203 typ srflx raddr 192.168.1.7 rport 49203	[RFC5245]
a=ssrc:33333	[RFC5576]
cname:L/aoNWs11HmN4Xa5	
a=rtcp-rsize	[RFC5506]

Table 23: 5.2.10 SDP Answer

SDP Contents	RFC#/Notes

v=1	Version number incremented [RFC4566]
o=- 20518 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio video	[I-D.ietf-mmusic-sdp-bundle-negotiation] Alice wants to BUNDLE only audio and video media.
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 54609 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp:64678 IN IP4 24.23.204.141	[RFC3605]
a=mid:audio	[RFC5888]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=sendrecv	[RFC3264]
a=setup:actpass	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d: 1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
a=rtcp-fb:109 nack	[RFC5104]
a=ssrc:11111	[RFC5576]
cname:Q/NWs1ao1HmN4Xa5	
a=rtcp-rsize	[RFC5506]

m=video 54609 UDP/TLS/RTP/SAVPF 120	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtcp:64678 IN IP4 24.23.204.141	[RFC3605]
a=mid:video	[RFC5888]
a=rtpmap:120 VP8/90000	[I-D.ietf-payload-vp8]
a=sendrecv	[RFC3264]
a=setup:actpass	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
a=rtcp-fb:120 nack	[RFC5104]
a=rtcp-fb:120 nack pli	[RFC5104]
a=rtcp-fb:120 ccm fir	[RFC5104]
a=ssrc:22222 cname:Q/aoNWs11HmN4Xa5	[RFC5576]
a=rtcp-rsize	[RFC5506]

Table 24: 5.2.11 SDP Updated Offer

SDP Contents	RFC#/Notes
v=1	[RFC4566] Version number incremented
o=- 16833 0 IN IP4 0.0.0.0	[RFC4566] - Session Origin Information
s=-	[RFC4566]
t=0 0	[RFC4566]

a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio video	[I-D.ietf-mmusic-sdp-bundle-negotiation]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 49203 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp:60065 IN IP4 98.248.92.77	[RFC3605]
a=mid:audio	[RFC5888]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=sendrecv	[RFC3264]
a=setup:active	[RFC4145]
a=rtcp-mux	[RFC5761]
a=rtcp-fb:109 nack	[RFC5104]
a=ice-ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d4 7efbabd9a2	[RFC5245]
a=fingerprint:sha-1 99:41:49:83: 4a:97:0e:1f:ef:6d:f7:c9:c7:70: 9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.7 49203 typ host	[RFC5245]
a=candidate:1 1 UDP 1694302207 98.248.92.77 49203 typ srflx raddr 192.168.1.7 rport 49203	[RFC5245]
a=ssrc:33333 cname:L/aoNWs11HmN4Xa5	[RFC5576]
a=rtcp-rsize	[RFC5506]
m=video 49203 UDP/TLS/RTP/SAVPF 120	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtcp:60065 IN IP4 98.248.92.77	[RFC3605]
a=mid:video	[RFC5888]
a=rtpmap:120 VP8/90000	[I-D.ietf-payload-vp8]
a=sendrecv	[RFC3264]
a=setup:active	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ice-ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d4	[RFC5245]

7efbabd9a2	
a=fingerprint:sha-1 99:41:49:83: 4a:97:0e:1f:ef:6d:f7:c9:c7:70: 9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.7 49203 typ host	[RFC5245]
a=candidate:1 1 UDP 1694302207 98.248.92.77 49203 typ srflx raddr 192.168.1.7 rport 49203	[RFC5245]
a=rtcp-fb:120 nack	[RFC5104]
a=rtcp-fb:120 nack pli	[RFC5104]
a=rtcp-fb:120 ccm fir	[RFC5104]
a=ssrc:44444 cname:EocUG1f0fcg/yvY7	[RFC5576]
a=rtcp-rsize	[RFC5506]

Table 25: 5.2.11 SDP Updated Answer

5.3. MultiResolution, RTX, FEC Examples

This section deals with scenarios related to multi-source, multi-stream negotiation such as layered coding, simulcast, along with techniques that deal with providing robustness against transmission errors such as FEC and RTX. Also to note, mechanisms such as FEC and RTX could be envisioned in the above basic scenarios as well.

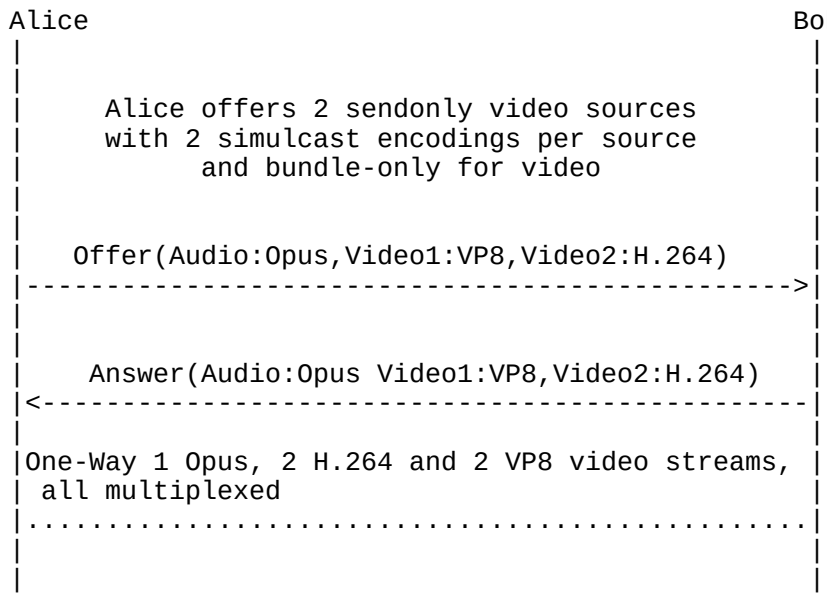
5.3.1. Sendonly Simulcast Session with 2 cameras and 2 encodings per camera

The SDP below shows Offer/Answer exchange with one audio and two video sources. Each of the video source can be sent at two different resolutions.

One video source corresponds to VP8 encoding, while the other corresponds to H.264 encoding.

bundle-only framework is used along with BUNDLE grouping framework to enable multiplexing of all the 5 streams (1 audio stream + 4 video streams) over a single RTP Session.

1 Way Successful Simulcast w/BUNDLE



SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE m0 m1 m2	[I-D.ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 54609 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp:64678 IN IP4 24.23.204.141	[RFC3605]
a=mid:m0	[RFC5888]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]

a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=sendonly	[RFC3264]
a=setup:actpass	[RFC4145]
a=rtcp-mux	[RFC5761]
a=rtcp-fb:109 nack	[RFC5104]
a=ssrc:11111 C90alEocUG1f0fcg	[RFC5576]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
a=rtcp-rsize	[RFC5506]
m=video 0 UDP/TLS/RTP/SAVPF 98 100	bundle-only video line with port number set to zero
c=IN IP4 24.23.204.141	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtcp:64678 IN IP4 24.23.204.141	[RFC3605]
a=mid:m1	[RFC5888] Video m=line part of BUNDLE group
a=rtpmap:98 VP8/90000	[I-D.ietf-payload-vp8]
a=rtpmap:100 VP8/90000	[I-D.ietf-payload-vp8]
a=imageattr:98 [x=1280,y=720]	[RFC6236]Camera-1,Encoding-1 Resolution
a=fmtp:98 max-fr=30	[RFC4566]
a=imageattr:100 [x=640,y=480]	[RFC6236] Camera-1,Encoding-2 Resolution
a=fmtp:100 max-fr=15	[RFC4566]
a=simulcast: send 98;100	[I-D.ietf-mmusic-sdp-simulcast] Alice can send 2 resolutions
a=ssrc:12345 cname:axzo1278npDlAzM73	[RFC5576] [RFC7022] Camera-1,Encoding-1 SSRC

	with Session CNAME
a=ssrc:45678 cname:axzo1278npDlAzM73	[RFC5576] [RFC7022] Camera-1,Encoding-2 SSRC with Session CNAME
a=sendonly	[RFC3264] - Send only video stream
a=rtcp-mux	[RFC5761]
a=bundle-only	[UNIFIED-PLAN]
a=rtcp-fb:98 nack	[RFC5104]
a=rtcp-fb:98 nack pli	[RFC5104]
a=rtcp-fb:98 ccm fir	[RFC5104]
a=rtcp-fb:100 nack	[RFC5104]
a=rtcp-fb:100 nack pli	[RFC5104]
a=rtcp-fb:100 ccm fir	[RFC5104]
a=rtcp-rsize	[RFC5506]
m=video 0 UDP/TLS/RTP/SAVPF 101 102	bundle-only video line with port number set to zero
c=IN IP4 24.23.204.141	[RFC4566]
a=msid:ma tc	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tc)
a=rtcp:64678 IN IP4 24.23.204.141	[RFC3605]
a=mid:m2	[RFC5888] Video m=line part of BUNDLE group
a=rtpmap:101 H264/90000	[RFC3984]
a=rtpmap:102 H264/90000	[RFC3984]
a=fmtp:101 profile-level-id=4d0028 ;packetization-mode=1;max-fr=30	[RFC3984]Camera-2,Encoding-1 Resolution
a=fmtp:102 profile-level-id=4d0028 ;packetization-mode=1;max-fr=15	[RFC3984]Camera-2,Encoding-2 Resolution
a=simulcast: send 101;102	[I-D.ietf-mmusic-sdp-simulcast]
a=ssrc:67890 cname:axzo1278npDlAzM73	[RFC5576] [RFC7022] Camera-2,Encoding-1 SSRC with Session CNAME
a=ssrc:56789 cname:axzo1278npDlAzM73	[RFC5576] [RFC7022] Camera-2,Encoding-2 SSRC with Session CNAME
a=sendonly	[RFC3264] - Send only video stream
a=rtcp-mux	[RFC5761]
a=bundle-only	[UNIFIED-PLAN]
a=rtcp-fb:101 nack	[RFC5104]
a=rtcp-fb:101 nack pli	[RFC5104]
a=rtcp-fb:101 ccm fir	[RFC5104]
a=rtcp-fb:102 nack	[RFC5104]
a=rtcp-fb:102 nack pli	[RFC5104]
a=rtcp-fb:102 ccm fir	[RFC5104]

a=rtcp-rsize	[RFC5506]

Table 26: 5.3.1 SDP Offer

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE m0 m1 m2	[I-D.ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 49203 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=rtcp:60065 IN IP4 98.248.92.77	[RFC3605]
a=mid:m0	[RFC5888]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=rtcp-fb:109 nack	[RFC5104]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=recvonly	[RFC3264]
a=setup:active	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ssrc:22222 cname:y8/C90alEocUG1f0fcg	[RFC5576]
a=ice-ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 2 UDP 2113667327 192.168.1.7 49203 typ host	[RFC5245]
a=candidate:1 2 UDP 694302207 98.248.92.77 49203 typ srflx raddr 192.168.1.4 rport 49203	[RFC5245]

a=rtcp-rsize	[RFC5506]
m=video 49203 UDP/TLS/RTP/SAVPF 98 100	BUNDLE accepted with port repeated from the audio port
c=IN IP4 98.248.92.77	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtcp:60065 IN IP4 98.248.92.77	[RFC3605]
a=mid:m1	[RFC5888] Video m=line part of BUNDLE group
a=rtpmap:98 VP8/90000	[I-D.ietf-payload-vp8]
a=rtpmap:100 VP8/90000	[I-D.ietf-payload-vp8]
a=imageattr:98 [x=1280,y=720]	[RFC6236]Camera-1,Encoding-1 Resolution
a=fmtp:98 max-fr=30	[RFC4566]
a=imageattr:100 [x=640,y=480]	[RFC6236] Camera-1,Encoding-2 Resolution
a=fmtp:100 max-fr=15	[RFC4566]
a=recvonly	[RFC3264] - receive only video stream
a=simulcast: recv 98;100	[I-D.ietf-mmusic-sdp-simulcast]
a=ssrc:54321 cname:y8/C90alEocUG1f0fcg	[RFC5576]
a=ice-ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 60065 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 98.248.92.77 60065 typ srflx raddr 192.168.1.4 rport 60065	[RFC5245]
a=setup:active	[RFC4145]
a=rtcp-mux	[RFC5576]
a=bundle-only	[UNIFIED-PLAN]
a=rtcp-rsize	[RFC5506]
m=video 54609 UDP/TLS/RTP/SAVPF 101 102	BUNDLE accepted with port repeated from the audio port
c=IN IP4 98.248.92.77	[RFC4566]
a=rtcp:56503 IN IP4 98.248.92.77	[RFC3605]
a=msid:ma tc	Identifies RTCMediaStream ID (ma) and

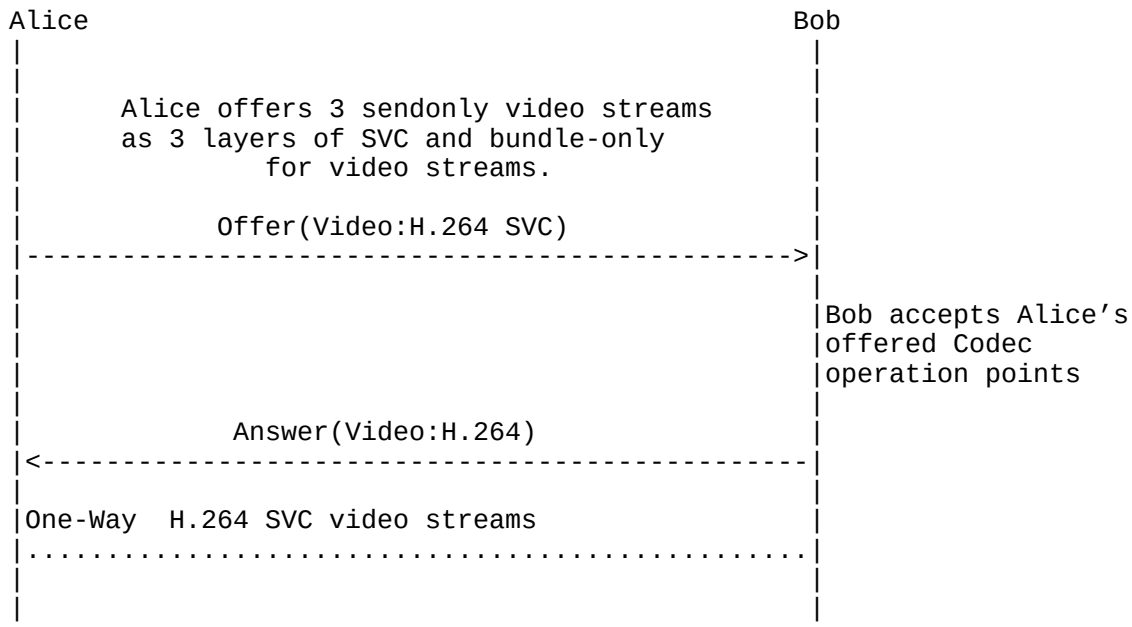
		RTCMediaStreamTrack ID (tc)
a=mid:m2		[RFC5888] Video m=line part of BUNDLE group
a=rtpmap:101 H264/90000		[RFC3984]
a=rtpmap:102 H264/90000		[RFC3984]
a=recvonly		[RFC3264]
a=fmtp:101 profile-level-id=4d0028 ;packetization-mode=1;max-fr=30		[RFC3984]
a=fmtp:102 profile-level-id=4d0028 ;packetization-mode=1;max-fr=15		[RFC3984]
a=simulcast: recv 101;102		[I-D.ietf-mmusic-sdp-simulcast] Bob accepts to receieve the offered simulcast streams
a=ssrc:90876 cname:axzo1278npDlAzM73		[RFC5576]
a=ice-ufrag:ufrag:c300d85b		[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d47ef babd9a2		[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a: 97:0e:1f:ef:6d:f7:c9:c7:70: 9d:1f:66:79:a8:07		[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 60065 typ host		[RFC5245]
a=candidate:1 2 UDP 1694302206 98.248.92.77 60065 typ srflx raddr 192.168.1.7 rport 60065		[RFC5245]
a=setup:active		[RFC4145]
a=rtcp-mux		[RFC5576]
a=bundle-only		[UNIFIED-PLAN]
a=rtcp-rsize		[RFC5506]

Table 27: 5.3.1 SDP Anwer

5.3.2. Successful SVC Video Session

This section shows an SDP Offer/Answer for a session with an audio and a single video source. The video source is encoded as layered coding at 3 different resolutions based on [RFC5583]. The video m=line shows 3 streams with last stream (payload 100) dependent on streams with payload 96 and 97 for decoding.

SVC Session - 3 Layers w/BUNDLE



SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE m0 m1	[I-D.ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 54609 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp:64678 IN IP4 24.23.204.141	[RFC3605]
a=mid:m0	[RFC5888] Audio m=line part of

	BUNDLE group with a unique port number
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=sendonly	[RFC3264]
a=rtcp-fb:109 nack	[RFC5104]
a=setup:actpass	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
a=ssrc:67890 cname:axzo1278npDlAzM73	[RFC5576]
a=rtcp-rsize	[RFC5506]
m=video 0 UDP/TLS/RTP/SAVPF 96 97 100	bundle-only video line with port number set to zero
c=IN IP4 24.23.204.141	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tc)
a=rtcp:64678 IN IP4 24.23.204.141	[RFC3605]
a=mid:m1 a=msid:ma tb	[RFC5888] Audio m=line part of BUNDLE group
a=rtpmap:96 H264/90000	[RFC3984]
a=fmtp:96 profile-level-id=4d0028; packetization-mode=1;max-fr=30;max-fs=8040	[RFC3984]H.264 Layer 1
a=rtpmap:97 H264/90000	[RFC3984]
a=fmtp:97 profile-level-id=4d0028;packetization-mode=1; max-fr=15;max-fs=1200	[RFC3984] H.264 Layer 2

a=rtpmap:100 H264-SVC/90000	[RFC3984]
a=fmtp:100 profile-level-id=4d0028;packetization-mode=1; max-fr=30;max-fs=8040	[RFC3984]
a=depend:100 lay m1:96,97;	[RFC5583]Layer 3 dependent on layers 1 and 2
a=sendonly	[RFC3264] - Send only video stream
a=rtcp-mux	[RFC5761]
a=bundle-only	[UNIFIED-PLAN]
a=ssrc:1732846380 cname:axzo1278npDlAzM73	[RFC5576]
a=ssrc:1732846381 cname:axzo1278npDlAzM73	[RFC5576]
a=ssrc:1732846382 cname:axzo1278npDlAzM73	[RFC5576]
a=rtcp-fb:* nack	[RFC5104]
a=rtcp-fb:* nack pli	[RFC5104]
a=rtcp-fb:* ccm fir	[RFC5104]
a=rtcp-rsize	[RFC5506]

Table 28: 5.3.2 SDP Offer with SVC

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE m0 m1	[I-D.ietf-mmusic-sdp-bundle-negotiation]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 49203 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp:60065 IN IP4 98.248.92.77	[RFC3605]
a=mid:m0	[RFC5888]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=rtcp-fb:109 nack	[RFC5104]

a=recvonly	[RFC3264]
a=setup:active	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3 474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83: 4a:97:0e:1f:ef:6d:f7:c9:c7:70: 9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 60065 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 98.248.92.77 60065 typ srflx raddr 192.168.1.5 rport 60065	[RFC5245]
a=rtcp-rsize	[RFC5506]
m=video 54609 UDP/TLS/RTP/SAVPF 96 100	BUNDLE accepted Bundle address same as audio m=line.
c=IN IP4 98.248.92.77	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtcp:56503 IN IP4 98.248.92.77	[RFC3605]
a=mid:m1	[RFC5888] Video m=line part of BUNDLE group
a=rtpmap:96 H264/90000	[RFC3984]
a=fmtp:96 profile-level-id=4d0028;packetization-mode=1; max-fr=30;max-fs=8040	[RFC3984]H.264 Layer 1
a=rtpmap:100 H264-SVC/90000	[RFC3984]
a=fmtp:100 profile-level-id=4d0028;packetization-mode=1; max-fr=30;max-fs=8040	[RFC3984]
a=depend:100 lay m1:96;	[RFC5583] Bob chooses 2 Codec Operation points
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3 474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83: 4a:97:0e:1f:ef:6d:f7:c9:c7:70: 9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.5 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.142 64678 typ srflx raddr 192.168.1.5 rport 64678	[RFC5245]
a=recvonly	[RFC3264] - Receive only video stream
a=setup:active	[RFC4145]

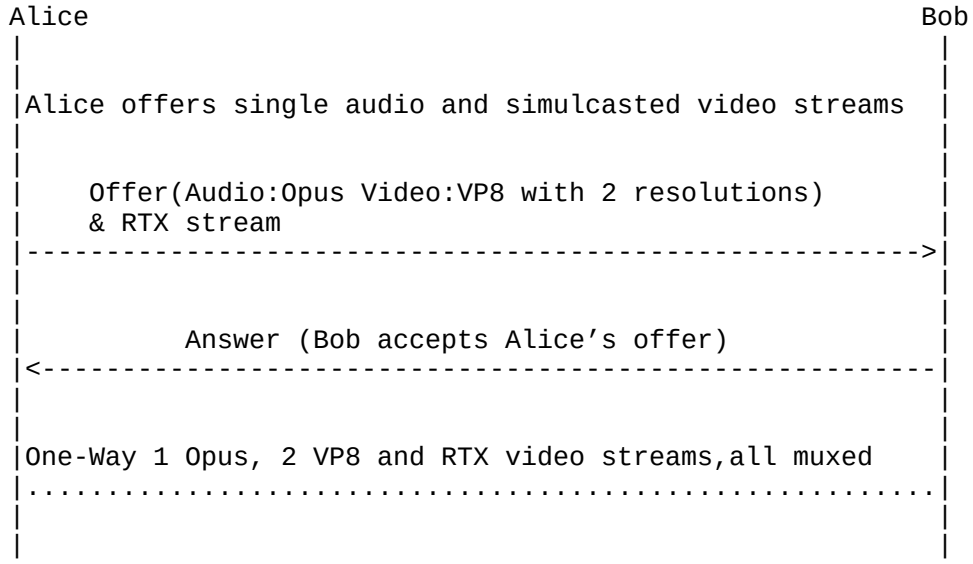
a=rtcp-mux	[RFC5761]
a=bundle-only	[UNIFIED-PLAN]
a=ssrc:4638117328	[RFC5576]
cname:axzo1278npDlAzM73	
a=rtcp-rsize	[RFC5506]

Table 29: 5.3.2 SDP Answer with SVC

5.3.3. Successful Simulcast Video Session with Retransmission

This section shows an SDP Offer/Answer exchange for a simulcast scenario with 2 two resolutions and has [RFC4588] style re-transmission flows.

Simulcast Streams with Retransmission



SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]

a=group:BUNDLE m0 m1	[I-D.ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 54609 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp:64678 IN IP4 24.23.204.141	[RFC3605]
a=mid:m0	[RFC5888] Audio m=line part of BUNDLE group with a unique port number
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=rtcp-fb:109 nack	[RFC5104]
a=sendonly	[RFC3264]
a=setup:actpass	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ssrc:11111	[RFC5576]
cname:EocUG1f0fcg/yvY7	
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
a=rtcp-rsize	[RFC5506]
m=video 0 UDP/TLS/RTP/SAVPF 98 100 101 103	bundle-only video line with port number set to zero
c=IN IP4 24.23.204.141	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack

		ID (tb)
a=rtcp:64678 IN IP4		[RFC3605]
24.23.204.141		
a=mid:m1		[RFC5888]
a=rtpmap:98 VP8/90000		[I-D.ietf-payload-vp8]
a=rtpmap:100 VP8/90000		[I-D.ietf-payload-vp8]
a=rtpmap:101 VP8/90000		[I-D.ietf-payload-vp8]
a=rtpmap:103 VP8/90000		[I-D.ietf-payload-vp8]
a=fmtp:98 max-fr=30;max-fs=8040		[RFC4566]
a=fmtp:100 max-fr=15;max-fs=1200		[RFC4566]
a=fmtp:101 apt=98;rtx-time=3000		[RFC4588]
a=fmtp:103 apt=100;rtx-time=3000		[RFC4588]
a=simulast: send 98;100		[I-D.ietf-mmusic-sdp-simulcast]
		]
a=ssrc-group:FID 12345 34567		[RFC5888]
a=ssrc-group:FID 78990 90887		[RFC5888]
a=ssrc:12345		[RFC5576]
cname:Q/NWs1ao1HmN4Xa5		
a=ssrc:78990		[RFC5576]
cname:Q/NWs1ao1HmN4Xa5		
a=ssrc:34567		[RFC5576]
cname:Q/NWs1ao1HmN4Xa5		
a=ssrc:90887		[RFC5576]
cname:Q/NWs1ao1HmN4Xa5		
a=sendonly		[RFC3264]
a=rtcp-mux		[RFC5761]
a=bundle-only		[UNIFIED-PLAN]
a=rtcp-fb:* nack		[RFC5104]
a=rtcp-fb:* nack pli		[RFC5104]
a=rtcp-fb:* ccm fir		[RFC5104]
a=rtcp-rsize		[RFC5506]

Table 30: 5.3.3 SDP Offer w/Simulcast, RTX

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE m0 m1	[I-D.ietf-mmusic-sdp-bundle-ne gotiation] Alice supports grouping of m=lines under BUNDLE semantics
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]

m=audio 49203 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp:60065 IN IP4 98.248.92.77	[RFC3605]
a=mid:m0	[RFC5888]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=rtcp-fb:109 nack	[RFC5104]
a=recvonly	[RFC3264]
a=setup:active	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ssrc:33333 cname:L/HmN4Xa5NWs1ao1	[RFC5576]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 98.248.92.77 64678 typ srflx raddr 192.168.1.7 rport 60065	[RFC5245]
a=rtcp-rsize	[RFC5506]
m=video 49203 UDP/TLS/RTP/SAVPF 98 100 101 103	BUNDLE accepted with Bundle address identical to audio m-line
c=IN IP4 98.248.92.77	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtcp:60065 IN IP4 98.248.92.77	[RFC3605]
a=mid:m1	[RFC5888] Video m=line part of BUNDLE group
a=rtpmap:98 VP8/90000	[I-D.ietf-payload-vp8]
a=rtpmap:100 VP8/90000	[I-D.ietf-payload-vp8]
a=rtpmap:101 VP8/90000	[I-D.ietf-payload-vp8]
a=rtpmap:103 VP8/90000	[I-D.ietf-payload-vp8]
a=fmtp:98 max-fr=30;max-fs=8040	[RFC4566]
a=fmtp:100 max-fr=15;max-fs=1200	[RFC4566]
a=fmtp:101 apt=98;rtx-time=3000	[RFC4588]
a=fmtp:103 apt=100;rtx-time=3000	[RFC4588]

a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3 474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83: 4a:97:0e:1f:ef:6d:f7:c9:c7: 70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 60065 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 98.248.92.772 60065 typ srflx raddr 192.168.1.7 rport 60065	[RFC5245]
a=simulcast: recv 98;100	[I-D.ietf-mmusic-sdp-simulcast]
a=recvonly	[RFC3264]
a=setup:active	[RFC4145]
a=rtcp-mux	[RFC5761]
a=bundle-only	[UNIFIED-PLAN]
a=rtcp-fb:* nack	[RFC5104]
a=rtcp-fb:* nack pli	[RFC5104]
a=rtcp-fb:* ccm fir	[RFC5104]
a=rtcp-rsize	[RFC5506]

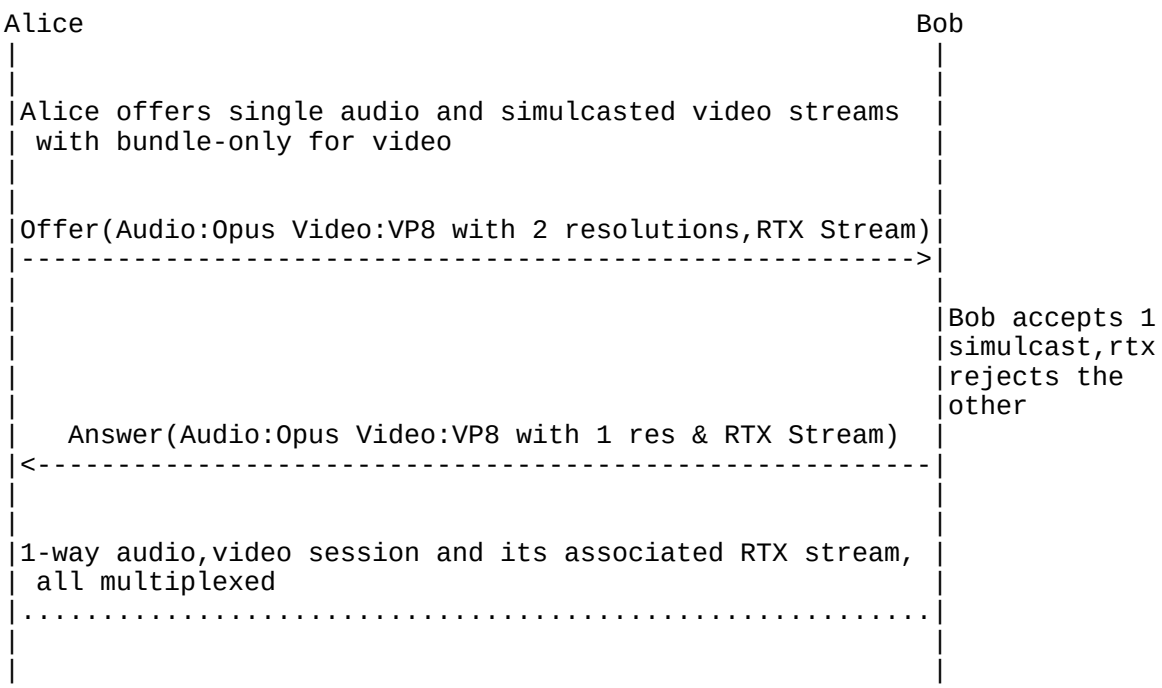
Table 31: 5.3.3 SDP Answer w/Simulcast, RTX

5.3.4. Successful 1-way Simulcast Sessio with 2 resolutions and RTX - One resolution rejected

This section shows an SDP Offer/Answer exchange for a simulcast scenario with 2 two resolutions.

It also showcases when Bob rejects one of the Simulcast Video Stream which results in the rejection of the associated repair stream implicitly.

Simulcast Streams with Retransmission Rejected



SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE m0 m1	[I-D.ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 54609 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp:64678 IN IP4	[RFC3605]

24.23.204.141	
a=mid:m0	[RFC5888]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=rtcp-fb:109 nack	[RFC5104]
a=sendonly	[RFC3264]
a=setup:actpass	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ssrc:11111 cname:LP/NWs1ao1HmN4Xa5	[RFC5576]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
a=rtcp-rsize	[RFC5506]
m=video 0 UDP/TLS/RTP/SAVPF 98 100 101 103	bundle-only video line with port number set to zero
c=IN IP4 24.23.204.141	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtcp:64678 IN IP4 24.23.204.141	[RFC3605]
a=mid:m1	[RFC5888]
a=rtpmap:98 VP8/90000	[I-D.ietf-payload-vp8]
a=rtpmap:100 VP8/90000	[I-D.ietf-payload-vp8]
a=rtpmap:101 VP8/90000	[I-D.ietf-payload-vp8]
a=rtpmap:103 VP8/90000	[I-D.ietf-payload-vp8]
a=fmtp:98 max-fr=30;max-fs=8040	[RFC4566]
a=fmtp:100 max-fr=15;max-fs=1200	[RFC4566]
a=fmtp:101 apt=98;rtx-time=3000	[RFC4588]
a=fmtp:103 apt=100;rtx-time=3000	[RFC4588]
a=simulcast: send 98;100	[I-D.ietf-mmusic-sdp-simulcast]

a=ssrc-group:FID 12345 34567	[RFC5888]
a=ssrc-group:FID 78990 90887	[RFC5888]
a=ssrc:12345	[RFC5576]
cname:Q/NWs1ao1HmN4Xa5	
a=ssrc:78990	[RFC5576]
cname:Q/NWs1ao1HmN4Xa5	
a=ssrc:34567	[RFC5576]
cname:Q/NWs1ao1HmN4Xa5	
a=ssrc:90887	[RFC5576]
cname:Q/NWs1ao1HmN4Xa5	
a=sendonly	[RFC3264]
a=rtcp-mux	[RFC5761]
a=bundle-only	[UNIFIED-PLAN]
a=rtcp-fb:* nack	[RFC5104]
a=rtcp-fb:* nack pli	[RFC5104]
a=rtcp-fb:* ccm fir	[RFC5104]
a=rtcp-rsize	[RFC5506]

Table 32: 5.3.4 SDP Offer w/Simulcast, RTX

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE m0 m1	[I-D.ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 49203 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp:49203 IN IP4 98.248.92.77	[RFC3605]
a=mid:m0	[RFC5888]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=recvonly	[RFC3264]
a=setup:active	[RFC4145]

a=rtcp-mux	[RFC5761]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3 474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83: 4a:97:0e:1f:ef:6d:f7:c9:c7:70: 9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 60065 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 98.248.92.77 60065 typ srflx raddr 192.168.1.7 rport 60065	[RFC5245]
a=rtcp-rsize	[RFC5506]
m=video 49203 UDP/TLS/RTP/SAVPF 98 101	BUNDLE accepted with Bundle address identical to audio m-line
c=IN IP4 98.248.92.77	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtcp:60065 IN IP4 98.248.92.77	[RFC3605]
a=mid:m1	[RFC5888]
a=rtpmap:98 VP8/90000	[I-D.ietf-payload-vp8]
a=rtpmap:101 VP8/90000	[I-D.ietf-payload-vp8]
a=fmtp:98 max-fr=30;max-fs=8040	[RFC4566]
a=fmtp:101 apt=98;rtx-time=3000	[RFC4588]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3 474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83: 4a:97:0e:1f:ef:6d:f7:c9:c7:70: 9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 60065 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 98.248.92.77 60065 typ srflx raddr 192.168.1.5 rport 60065	[RFC5245]
a=simulcast: recv 98	[I-D.ietf-mmusic-sdp-simulcast] Bob accepts only one simulcast resolution
a=ssrc:54321 cname:NWs1ao1HmN4Xa5	[RFC5576]
a=recvonly	[RFC3264]
a=setup:active	[RFC4145]
a=rtcp-mux	[RFC5761]
a=bundle-only	[UNIFIED-PLAN]
a=rtcp-rsize	[RFC5506]

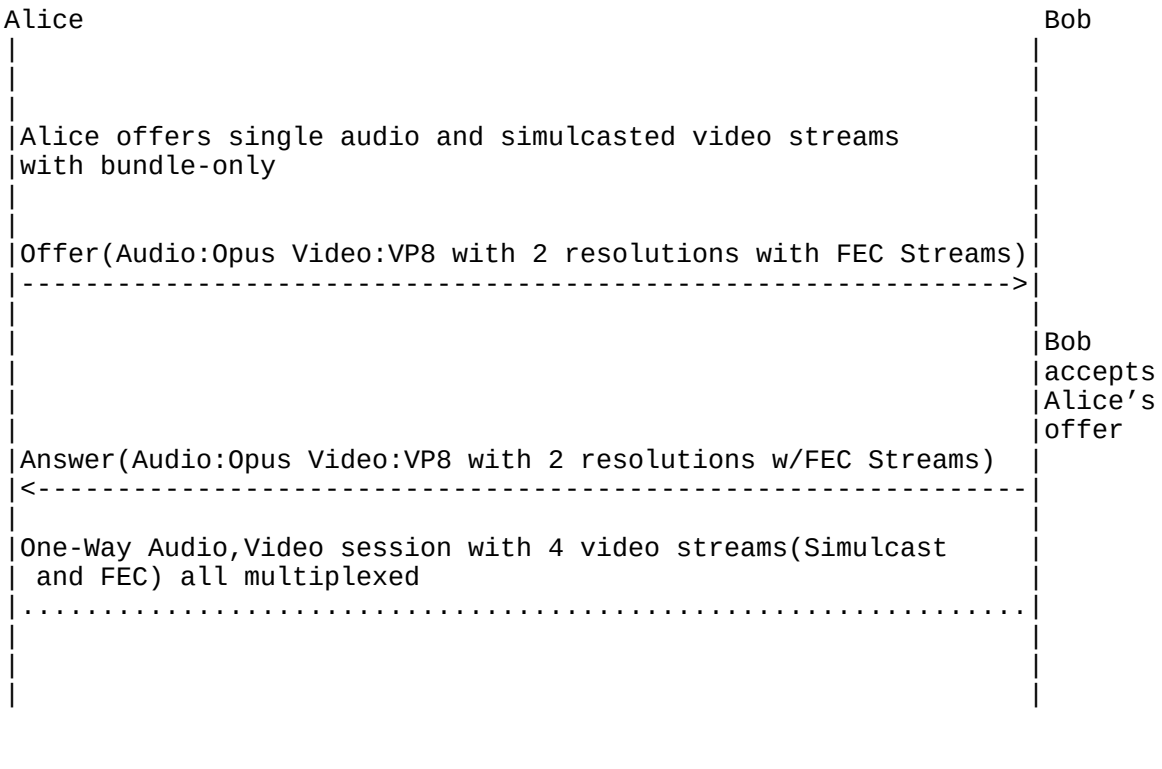
Table 33: 5.3.4 SDP Answer no Simulcast

5.3.5. Simulcast Video Session with Forward Error Correction

This section shows an SDP Offer/Answer exchange for Simulcast video stream at two resolutions and and has [RFC5956] style FEC flows.

On completion of the Offer/Answer exchange mechanism we end up one audio stream, 2 simulcast video streams and 2 associated FEC streams are sent over a single 5-tuple.

Simulcast Streams with Forward Error Correction



SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]

a=group:BUNDLE m0 m1	[I-D.ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 54609 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp:64678 IN IP4 24.23.204.141	[RFC3605]
a=mid:m0	[RFC5888]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=rtcp-fb:109 nack	[RFC5104]
a=sendonly	[RFC3264]
a=setup:actpass	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ssrc:11111 cname:Q/NWs1ao1HmN4Xa5	[RFC5576]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
a=rtcp-rsize	[RFC5506]
m=video 0 UDP/TLS/RTP/SAVPF 98 100 101 103	bundle-only video line with port number set to zero
c=IN IP4 24.23.204.141	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtcp:64678 IN IP4	[RFC3605]

24.23.204.141	
a=mid:m1	[RFC5888] Video m=line part of BUNDLE group
a=rtpmap:98 VP8/90000	[I-D.ietf-payload-vp8]
a=rtpmap:100 VP8/90000	[I-D.ietf-payload-vp8]
a=rtpmap:101 1d-interleaved-parityfec/90000	[RFC5956]
a=rtpmap:103 1d-interleaved-parityfec/90000	[RFC5956]
a=fmtp:98 max-fr=30;max-fs=8040	[RFC4566]
a=fmtp:100 max-fr=15;max-fs=1200	[RFC4566]
a=fmtp:101 L=5; D=10; repair-window=200000	[RFC5956]
a=fmtp:103 L=5; D=10; repair-window=200000	[RFC5956]
a=simulcast: send 98;100	[I-D.ietf-mmusic-sdp-simulcast]
a=depend:98 fec m1:101	TBD
a=depend:100 fec m1:103	TBD
a=ssrc-group:FEC-FR 12345 34567	[RFC5888]
a=ssrc-group:FEC-FR 78990 90887	[RFC5888]
a=ssrc:12345 cname:Q/NWs1ao1HmN4Xa5	[RFC5576]
a=ssrc:78990 cname:Q/NWs1ao1HmN4Xa5	[RFC5576]
a=ssrc:34567 cname:Q/NWs1ao1HmN4Xa5	[RFC5576]
a=ssrc:90887 cname:Q/NWs1ao1HmN4Xa5	[RFC5576]
a=sendonly	[RFC3264]
a=rtcp-mux	[RFC5761]
a=bundle-only	[UNIFIED-PLAN]
a=rtcp-fb:* nack	[RFC5104]
a=rtcp-fb:* nack pli	[RFC5104]
a=rtcp-fb:* ccm fir	[RFC5104]
a=rtcp-rsize	[RFC5506]

Table 34: 5.3.5 SDP Offer

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS m0	[I-D.ietf-mmusic-msid]

a=group:BUNDLE m0 m1	[I-D.ietf-mmusic-sdp-bundle-negotiation]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 49203 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp:60065 IN IP4 98.248.92.77	[RFC3605]
a=mid:m0	[RFC5888] Audio m=line part of BUNDLE group with a unique port number
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=rtcp-fb:109 nack	[RFC5104]
a=recvonly	[RFC3264]
a=setup:active	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ssrc:33333 cname:Y9/cZke09JAtpl98	[RFC5576]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d : 1f:66:79:a8:07	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 60065 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 98.248.92.77 60065 typ srflx raddr 192.168.1.7 rport 60065	[RFC5245]
a=rtcp-rsize	[RFC5506]
m=video 49203 UDP/TLS/RTP/SAVPF 98 100 101 103	BUNDLE accepted with Bundle Address identical to audio m=line.
c=IN IP4 98.248.92.77	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtcp:60065 IN IP4 98.248.92.77	[RFC3605]
a=mid:m1	[RFC5888] Video m=line part of BUNDLE group
a=rtpmap:98 VP8/90000	[I-D.ietf-payload-vp8]
a=rtpmap:100 VP8/90000	[I-D.ietf-payload-vp8]
a=rtpmap:101 1d-interleaved-	[RFC5956]

parityfec/90000	
a=rtpmap:103 1d-interleaved-parityfec/90000	[RFC5956]
a=fmtp:98 max-fr=30;max-fs=8040	[RFC4566]
a=fmtp:100 max-fr=15;max-fs=1200	[RFC4566]
a=fmtp:101 L=5; D=10; repair-window=200000	[RFC5956]
a=fmtp:103 L=5; D=10; repair-window=200000	[RFC5956]
a=simulcast: recv 98;100	[I-D.ietf-mmusic-sdp-simulcast]
a=depend:98 fec m1:101	TBD
a=depend:100 fec m1:103	TBD
a=recvonly	[RFC3264]
a=setup:active	[RFC4145]
a=rtcp-mux	[RFC5761]
a=bundle-only	[UNIFIED-PLAN]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 60065 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 98.248.92.77 60065 typ srflx raddr 192.168.1.7 rport 60065	[RFC5245]
a=rtcp-fb:* nack	[RFC5104]
a=rtcp-fb:* nack pli	[RFC5104]
a=rtcp-fb:* ccm fir	[RFC5104]
a=rtcp-rsize	[RFC5506]

Table 35: 5.3.5 SDP Answer

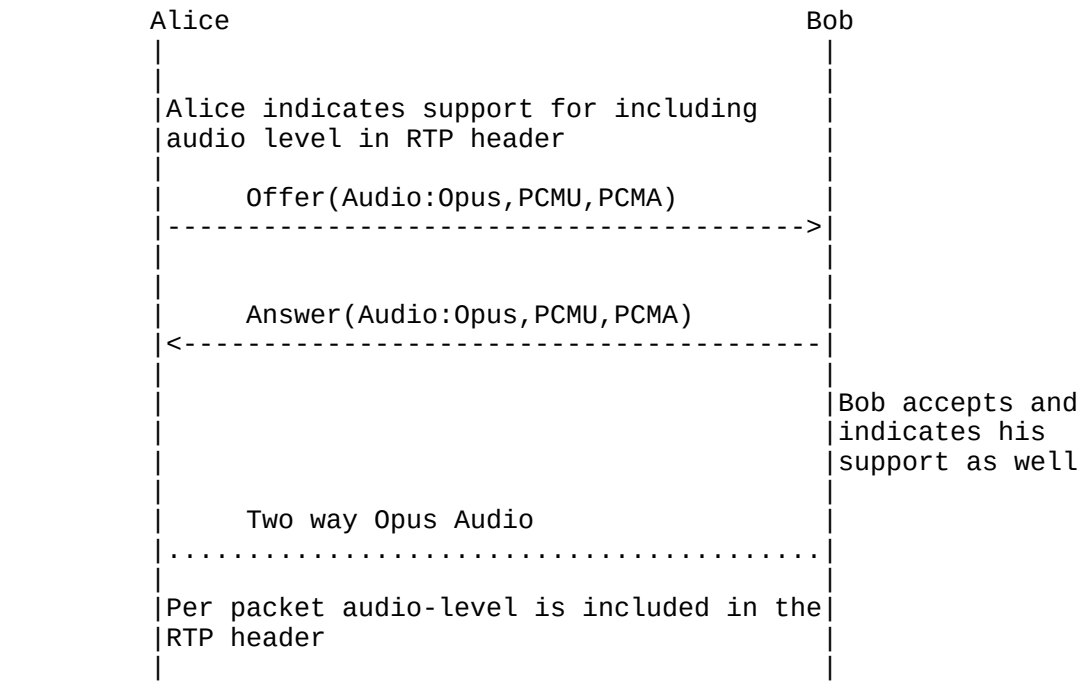
5.4. Others

The examples in the section provide SDP for a variety of scenarios related to RTP Header extension, Legacy Interop scenarios and more.

5.4.1. Audio Session - Voice Activity Detection

This example shows Alice indicating the support of the RTP header extension to include the audio-level of the audio sample carried in the RTP packet.

2-Way Audio with VAD



SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20518 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio	[I-D.ietf-mmusic-sdp-bundle-negotiation]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 54609 UDP/TLS/RTP/SAVPF 109 0 8	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=mid:audio	[RFC5888]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp:64678 IN IP4 24.23.204.141	[RFC3605]

a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=rtpmap:0 PCMU/8000	[RFC3551]
a=rtpmap:0 PCMA/8000	[RFC3551]
a=sendrecv	[RFC3264]
a=setup:actpass	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
a=rtcp-fb:* nack	[RFC5104]
a=ssrc:11111 cname:QCL/1HmN4Xa5CClapa	[RFC5576]
a=rtcp-rsize	[RFC5506]

Table 36: 5.4.1 SDP Offer

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 16833 0 IN IP4 0.0.0.0	[RFC4566] - Session Origin Information
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio	[I-D.ietf-mmusic-sdp-bundle-negotiation]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 49203 UDP/TLS/RTP/SAVPF 109 0 98	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]

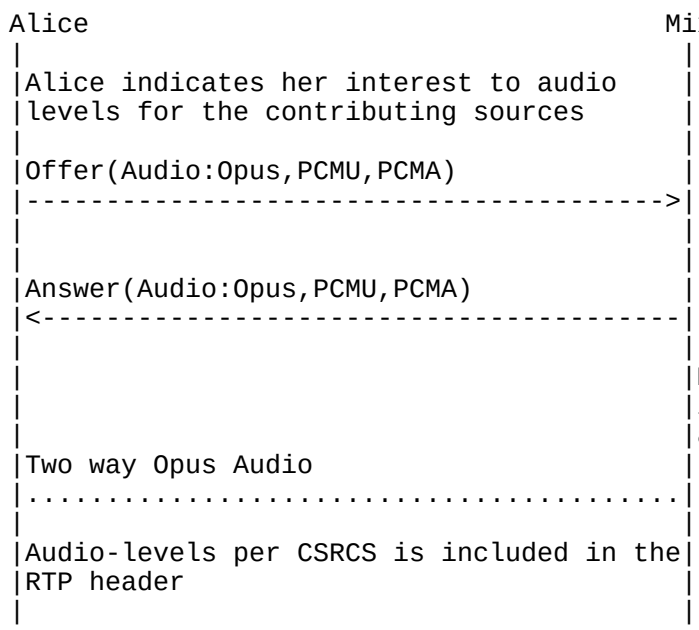
a=mid:audio	[RFC5888]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp:60065 IN IP4 98.248.92.77	[RFC3605]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus] - Bob accepts only Opus Codec
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=rtpmap:0 PCMU/8000	[RFC3551] PCMU Audio Codec
a=rtpmap:0 PCMA/8000	[RFC3551] PCMA Audio Codec
a=rtcp-fb:* nack	[RFC5104]
a=sendrecv	[RFC3264] - Bob can send and recv audio
a=setup:active	[RFC4145]
a=rtcp-mux	[RFC5761] - Bob can perform RTP/RTCP Muxing on port 49203
a=ice-ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.7 49203 typ host	[RFC5245]
a=candidate:1 1 UDP 1694302207 98.248.92.77 49203 typ srflx raddr 192.168.1.7 rport 49203	[RFC5245]
a=ssrc:1732846380 cname:EocUG1f0fcg/yvY7	[RFC5576]
a=rtcp-rsize	[RFC5506]

Table 37: 5.4.1 SDP Answer

5.4.2. Audio Conference - Voice Activity Detection

This example shows SDP for RTP header extension that allows RTP-level mixers in audio conferences to deliver information about the audio level of individual participants.

Audio Conference with VAD Support



SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20518 0 IN IP4 0.0.0.0	[RFC4566] - Session Origin Information
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio	[I-D.ietf-mmusic-sdp-bundle-negotiation]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 54609 UDP/TLS/RTP/SAVPF 109 0 8	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=mid:audio	[RFC5888]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp:64678 IN IP4 24.23.204.141	[RFC3605]
a=extmap:1/recvonly	[RFC6465]

urn:ietf:params:rtp-hdrext:csrc-audio-level	
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=rtpmap:0 PCMU/8000	[RFC3551] PCMU Audio Codec
a=rtpmap:0 PCMA/8000	[RFC3551] PCMA Audio Codec
a=rtcp-fb:* nack	[RFC5104]
a=sendrecv	[RFC3264] - Alice can send and recv audio
a=setup:actpass	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
a=ssrc:11111 cname:QCL/1HmN4Xa5CClapa	[RFC5576]
a=rtcp-rsize	[RFC5506]

Table 38: 5.4.2 SDP Offer

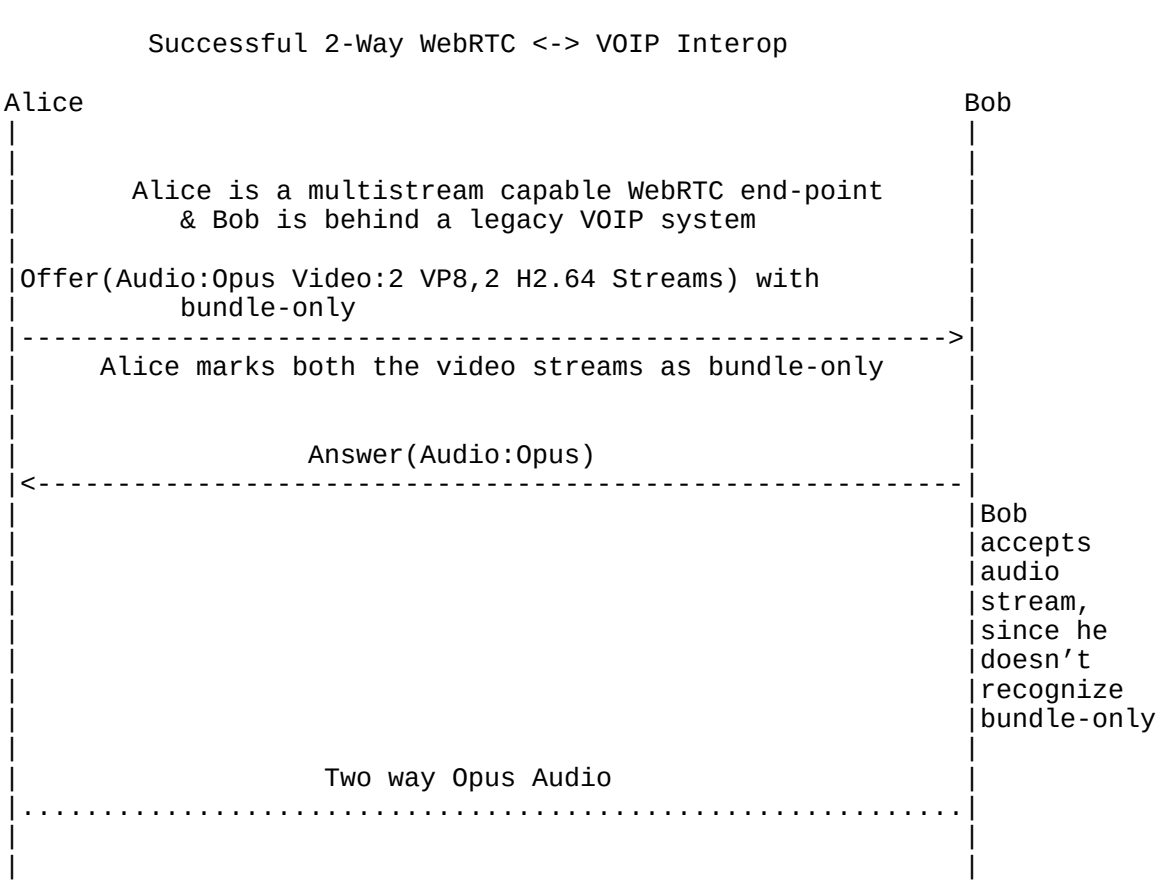
SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 16833 0 IN IP4 0.0.0.0	[RFC4566] - Session Origin Information
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE audio	[I-D.ietf-mmusic-sdp-bundle-negotiation]
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 49203 UDP/TLS/RTP/SAVPF 109 0 98	[RFC4566]
c=IN IP4 98.248.92.77	[RFC4566]
a=mid:audio	[RFC5888]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtcp:60065 IN IP4 98.248.92.77	[RFC3605]
a=extmap:1/sendonly urn:ietf:params:rtp-hdrext:csrc-audio-level	[RFC6465]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=rtpmap:0 PCMU/8000	[RFC3551] PCMU Audio Codec
a=rtpmap:0 PCMA/8000	[RFC3551] PCMA Audio Codec
a=rtcp-fb:* nack	[RFC5104]
a=sendrecv	[RFC3264]
a=setup:active	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ice-ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.7 49203 typ host	[RFC5245]
a=candidate:1 1 UDP 1694302207 98.248.92.77 49203 typ srflx raddr 192.168.1.7 rport 49203	[RFC5245]
a=ssrc:2222 cname:HmN4Xa5CC/lapa	[RFC5576]
a=rtcp-rsize	[RFC5506]

Table 39: 5.4.2 SDP Answer

5.4.3. Successful legacy Interop Fallaback with bundle-only

In the scenario described below, Alice is a multi-stream capable WebRTC endpoint while Bob is a legacy VOIP end-point. The SDP Offer/Answer exchange demonstrates successful session setup with fallback to audio only stream negotiated via bundle-only framework between the end-points. Specifically,

- Offer from Alice describes 2 cameras via 2 video m=lines with both marked as bundle-only.
- Since Bob doesnot recognize either the BUNDLE mechanism or the bundle-only attribute, he accepts only the audio stream from Alice.



SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=msid-semantic:WMS ma	[I-D.ietf-mmusic-msid]
a=group:BUNDLE m0 m1 m2	[I-D.ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
a=ice-options:trickle	[I-D.ietf-mmusic-trickle-ice]
m=audio 54609 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=rtcp:64678 IN IP4 24.23.204.141	[RFC3605]
a=mid:m0	[RFC5888] Audio m=line part of BUNDLE group with a unique port number
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=rtcp-fb:109 nack	[RFC5104]
a=sendrecv	[RFC3264]
a=setup:actpass	[RFC4145]
a=rtcp-mux	[RFC5761]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
a=ssrc:11111	[RFC5576]E

cname:axzo1278npDlAzM73	
a=rtcp-rsize	[RFC5506]
m=video 0 UDP/TLS/RTP/SAVPF 98 100	bundle-only video line with port number set to zero
c=IN IP4 24.23.204.141	[RFC4566]
a=rtcp:64678 IN IP4 24.23.204.141	[RFC3605]
a=mid:m1	[RFC5888] Video m=line part of BUNDLE group
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtpmap:98 VP8/90000	[I-D.ietf-payload-vp8]
a=imageattr:98 [x=1280,y=720]	[RFC6236]
a=fmtp:98 max-fr=30	[RFC4566]
a=ssrc:12345	[RFC5576]
cname:axzo1278npDlAzM73	
a=bundle-only	[UNIFIED-PLAN]
a=sendrecv	[RFC3264]
a=rtcp-rsize	[RFC5506]
m=video 0 UDP/TLS/RTP/SAVPF 101 103	bundle-only video line with port number set to zero
c=IN IP4 24.23.204.141	[RFC4566]
a=rtcp:64678 IN IP4 24.23.204.141	[RFC3605]
a=mid:m2	[RFC5888] Video m=line part of BUNDLE group
a=msid:ma tc	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tc)
a=rtpmap:101 H264/90000	[RFC3984]
a=rtpmap:103 H264/90000	[RFC3984]
a=fmtp:101 profile-level-id=4d0028 ;packetization-mode=1;max-fr=30	[RFC3984]Camera-2,Encoding-1 Resolution
a=ssrc:67890	[RFC5576]
cname:axzo1278npDlAzM73	
a=bundle-only	[UNIFIED-PLAN]
a=sendrecv	[RFC3264]
a=rtcp-rsize	[RFC5506]

Table 40: 5.4.3 SDP Simulcast bundle-only

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]

m=audio 49203 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=rtcp:60065 IN IP4 24.23.204.141	[RFC3605]
a=rtpmap:109 opus/48000/2	[I-D.ietf-payload-rtp-opus]
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level	[RFC6464]
a=ptime:20	[I-D.ietf-payload-rtp-opus]
a=rtcp-fb:109 nack	[RFC5104]
a=sendrecv	[RFC3264]
a=setup:active	[RFC4145]
a=ice-ufrag:ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=candidate:0 1 UDP 2113667327 192.168.1.7 49203 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 98.248.92.77 49203 typ srflx raddr 192.168.1.7 rport 49203	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 60065 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 98.248.92.77 60065 typ srflx raddr 192.168.1.7 rport 60065	[RFC5245]
a=rtcp-rsize	[RFC5506]
m=video 0 UDP/TLS/RTP/SAVPF 98 100	Bob doesn’t recognize bundle-only and hence rejects the video stream
c=IN IP4 98.248.92.77	[RFC4566]
a=rtpmap:98 VP8/90000	[I-D.ietf-payload-vp8]
a=rtpmap:100 VP8/90000	[I-D.ietf-payload-vp8]
a=imageattr:98 [x=1280,y=720]	[RFC6236]
a=fmtp:98 max-fr=30	[RFC4566]
m=video 0 UDP/TLS/RTP/SAVPF 98 100	Bob doesn’t recognize bundle-only and hence rejects the video stream
c=IN IP4 98.248.92.77	[RFC4566]
a=rtpmap:101 H264/90000	[RFC3984]
a=fmtp:101 profile-level-id=4d0028;packetization-mode=1;max-fr=30	[RFC3984]Camera-2,Encoding-1 Resolution

Table 41: 5.4.3 SDP Anwer

5.4.4. Legacy Interop with RTP/AVP profile

In this section, we attempt to provide session descriptions showcasing inter-operability between a WebRTC end-point and a Legacy VOIP end-point. The ideas included in here are not fully baked into the standards and might be controversial in nature. The hope here is to demonstrate a plausible SDP composition to enchance seamless inter-operability between the aforementioned communication systems.

In the scenario desribed below, Alice is a legacy end-point which sends [RFC3264] Offer with two sets of media descriptions per media type.

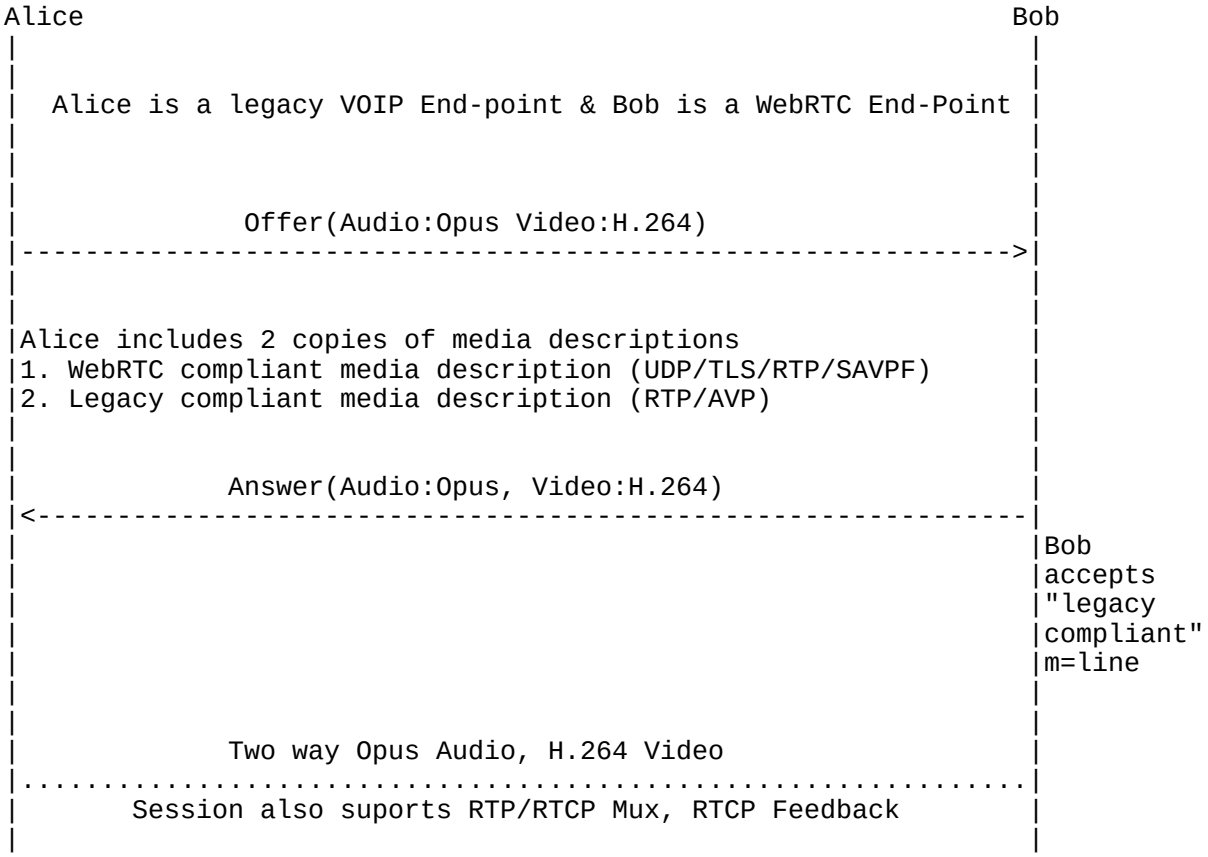
One set that correponds to [WebRTC] compliant UDP/TLS/RTP/SAVPF based audio and video descriptions.

Another set with RTP/AVP based audio and video descriptions for the legacy Interop purposes.

Also to note, Alice includes session level DTLS information and media level RTCP feedback information as applicable to both the sets of media descriptions

On the other hand, Bob being a WebRTC end-point, recognizes accepts the media descriptions with RTP/AVP profile. The security and feedback requirements for the session are either handled by a intermediate gateway or with some combination of Alice’s capabilities and the intermediate gateway.

Successful 2-Way WebRTC <-> VOIP Interop



SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 20518 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:e f:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=rtcp-rsize	[RFC5506]
m=audio 54609 UDP/TLS/RTP/SAVPF 109	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=rtpmap:109 opus/48000	

a=ptime:20	
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
a=rtcp-fb:109 nack	[RFC5104]
m=video 62537 UDP/TLS/RTP/SAVPF 120	[RFC4566]
c=IN IP4 24.23.204.141	[RFC4566]
a=rtpmap:120 VP8/90000	[I-D.ietf-payload-vp8]
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=candidate:0 1 UDP 2113667327 192.168.1.4 62537 typ host	[RFC5245]
a=candidate:1 1 UDP 1694302207 24.23.204.141 62537 typ srflx raddr 192.168.1.4 rport 62537	[RFC5245]
a=candidate:0 2 2113667326 192.168.1.4 54721 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 54721 typ srflx raddr 192.168.1.4 rport 54721	[RFC5245]
a=rtcp-fb:120 nack pli	[RFC5104]
a=rtcp-fb:120 ccm fir	[RFC5104]
---	These set of media descriptions are for Legacy Inter-op purposes
m=audio 54732 RTP/AVP 109	[RFC4566]Alice includes RTP/AVP audio stream description
c=IN IP4 24.23.204.141	[RFC4566]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:7f:7d:f9:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=rtpmap:109 opus/48000	
a=ptime:20	
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]Alice still includes RTP/RTCP Mux support
a=candidate:0 1 UDP 2113667327 192.168.1.4	[RFC5245]

54732 typ host a=candidate:1 1 UDP 694302207 24.23.204.141 54732 typ srflx raddr 192.168.1.4 rport 54732 a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245] [RFC5245] [RFC5245]
a=rtcp-fb:109 nack	[RFC5104]She adds her intent for NACK RTCP feedback support
m=video 62445 RTP/AVP 120	[RFC4566]Alice includes RTP/AVP video stream description
c=IN IP4 24.23.204.141 a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:7d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC4566] [RFC5245]
a=rtpmap:120 VP8/90000	[I-D.ietf-payload-vp8]
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]Alice intends to perform RTP/RTCP Mux
a=candidate:0 1 UDP 2113667327 192.168.1.4 62445 typ host a=candidate:1 1 UDP 1694302207 24.23.204.141 62537 typ srflx raddr 192.168.1.4 rport 62445 a=candidate:0 2 2113667326 192.168.1.4 54721 typ host a=candidate:1 2 UDP 1694302206 24.23.204.141 54721 typ srflx raddr 192.168.1.4 rport 54721	[RFC5245] [RFC5245] [RFC5245] [RFC5245]
a=rtcp-fb:120 nack pli	[RFC5104] Alice indicates support for Picture loss Indication and NACK RTCP feedback
a=rtcp-fb:120 ccm fir	[RFC5104]

Table 42: 5.4.5 SDP Offer

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=- 16833 0 IN IP4 0.0.0.0	[RFC4566]
s=-	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:e f:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	[RFC5245]
m=audio 49203 RTP/AVP 109	[RFC4566] Bob accepts RTP/AVP based audio stream
c=IN IP4 98.248.92.77	[RFC4566]
a=rtpmap:109 opus/48000	
a=ptime:20	
a=sendrecv	[RFC3264]
a=candidate:0 1 UDP 2113667327 192.168.1.7 49203 typ host	[RFC5245]
a=candidate:1 1 UDP 1694302207 98.248.92.77 49203 typ srflx raddr 192.168.1.7 rport 49203	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 60065 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 98.248.92.77 60065 typ srflx raddr 192.168.1.7 rport 60065	[RFC5245]
m=video 63130 RTP/SAVP 120	[RFC4566] Bob accepts RTP/AVP based video stram
c=IN IP4 98.248.92.771	[RFC4566]
a=rtpmap:120 VP8/90000	[I-D.ietf-payload -vp8]
a=sendrecv	[RFC3264]
a=candidate:0 1 UDP 2113667327 192.168.1.7 63130 typ host	[RFC5245]
a=candidate:1 1 UDP 1694302207 98.248.92.77 63130 typ srflx raddr 192.168.1.7 rport 63130	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 56607 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 98.248.92.77 56607 typ srflx raddr 192.168.1.7 rport 56607	[RFC5245]

Table 43: 5.4.5 SDP Answer

6. IANA Considerations

This document requires no actions from IANA.

7. Acknowledgments

We would like to thanks Justin Uberti, Chris Flo for their detailed review and inputs.

8. Change Log

[RFC EDITOR NOTE: Please remove this section when publishing]

Changes from draft-nandakumar-rtcweb-sdp-06 and draft-nandakumar-rtcweb-sdp-07

- Added clarification on Call-Flow diagram usage
- More cleanups

Changes from draft-nandakumar-rtcweb-sdp-05

- Added Ascii chart for all the SDP Eaxamples
- Improved text and updated SDP Examples for Simulcast and FEC
- Fixed MediaStream ID Semantics SDP Errors

Changes from draft-nandakumar-rtcweb-sdp-04

- Interim version of the draft to avert expiry
- Corrected placement of c= line as per RFC4566
- Updated simulcast SDP to reflect draft-westerlund-avtcore-rtp-simulcast-04

Changes from draft-nandakumar-rtcweb-sdp-03

- Aligned more closely with JSEP version -05
- Added Conventions to help readability
- Add more examples to clarify BUNDLE use-cases

Changes from draft-nandakumar-rtcweb-sdp-02

- Major refactoring was done to group the examples in to categories

- SDP was updated through out to reflect JSEP-04 style of defining attributes per m=line than at the session level.
- Added 8 new examples.
- Updated references for Trickle, Unified Plan
- Add section to explain the syntax conventions followed in the examples.

Changes from draft-nandakumar-rtcweb-sdp-01

- Updated references to OPUS RTP Payload Specification.
- Updated BUNDLE examples based on the latest draft-ietf-mmusic-sdp-bundle-negotiation.
- Added examples for multiple audio and video flows based on Unified Plan.
- Added new examples for RTX and FEC streams
- Updated Simulcast and SVC examples

Changes from draft-nandakumar-rtcweb-sdp-00

- Fixed editorial comments on the mailing list.
- Updated Data-channel SDP information based on draft-ietf-mmusic-sctp-sdp.
- Updated BUNDLE examples based on draft-ietf-mmusic-sdp-bundle-negotiation.
- Added examples for few more BUNDLE variants
- Added new examples for Simulcast and SVC

9. Informative References

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC4145] Yon, D. and G. Camarillo, "TCP-Based Media Transport in the Session Description Protocol (SDP)", RFC 4145, September 2005.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.

[RFC3952] Duric, A. and S. Andersen, "Real-time Transport Protocol (RTP) Payload Format for internet Low Bit Rate Codec (iLBC) Speech", RFC 3952, December 2004.

[RFC4796] Hautakorpi, J. and G. Camarillo, "The Session Description Protocol (SDP) Content Attribute", RFC 4796, February 2007.

[RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.

[RFC3556] Casner, S., "Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth", RFC 3556, July 2003.

[RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, February 2008.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.

[RFC5956] Begen, A., "Forward Error Correction Grouping Semantics in the Session Description Protocol", RFC 5956, September 2010.

[RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.

[RFC6236] Johansson, I. and K. Jung, "Negotiation of Generic Image Attributes in the Session Description Protocol (SDP)", RFC 6236, May 2011.

[RFC3984] Wenger, S., Hannuksela, M., Stockhammer, T., Westerlund, M., and D. Singer, "RTP Payload Format for H.264 Video", RFC 3984, February 2005.

[RFC5583] Schierl, T. and S. Wenger, "Signaling Media Decoding Dependency in the Session Description Protocol (SDP)", RFC 5583, July 2009.

[RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

[RFC3605] Huitema, C., "Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP)", RFC 3605, October 2003.

[RFC2833] Schulzrinne, H. and S. Petrack, "RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals", RFC 2833, May 2000.

[RFC6464] Lennox, J., Ivov, E., and E. Marocco, "A Real-time Transport Protocol (RTP) Header Extension for Client-to-Mixer Audio Level Indication", RFC 6464, December 2011.

[RFC6465] Ivov, E., Marocco, E., and J. Lennox, "A Real-time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication", RFC 6465, December 2011.

[RFC7022] Begen, A., Perkins, C., Wing, D., and E. Rescorla, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 7022, September 2013.

[I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-negotiation-12 (work in progress), October 2014.

[I-D.ietf-mmusic-sdp-simulcast]
Westerlund, M., Nandakumar, S., and M. Zanaty, "Using Simulcast in SDP and RTP Sessions", draft-ietf-mmusic-sdp-simulcast-00 (work in progress), January 2015.

[I-D.ietf-payload-rtp-opus]
Spittka, J., Vos, K., and J. Valin, "RTP Payload Format for Opus Speech and Audio Codec", draft-ietf-payload-rtp-opus-07 (work in progress), January 2015.

[I-D.ietf-payload-vp8]
Westin, P., Lundin, H., Glover, M., Uberti, J., and F. Galligan, "RTP Payload Format for VP8 Video", draft-ietf-payload-vp8-13 (work in progress), October 2014.

[I-D.ietf-rtcweb-jsep]
Uberti, J., Jennings, C., and E. Rescorla, "Javascript Session Establishment Protocol", draft-ietf-rtcweb-jsep-08 (work in progress), October 2014.

[I-D.ietf-mmusic-trickle-ice]
Ivov, E., Rescorla, E., and J. Uberti, "Trickle ICE: Incremental Provisioning of Candidates for the Interactive Connectivity Establishment (ICE) Protocol", draft-ietf-mmusic-trickle-ice-02 (work in progress), January 2015.

[I-D.ietf-mmusic-msid]
Alvestrand, H., "WebRTC MediaStream Identification in the Session Description Protocol", draft-ietf-mmusic-msid-07 (work in progress), October 2014.

[I-D.ietf-mmusic-sctp-sdp]
Holmberg, C., Loreto, S., and G. Camarillo, "Stream Control Transmission Protocol (SCTP)-Based Media Transport in the Session Description Protocol (SDP)", draft-ietf-mmusic-sctp-sdp-12 (work in progress), January 2015.

[I-D.ietf-rtcweb-data-channel]
Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", draft-ietf-rtcweb-data-channel-13 (work in progress), January 2015.

[WebRTC] W3C, "WebRTC 1.0: Real-time Communication Between Browsers", <http://dev.w3.org/2011/webrtc/editor/webrtc.html> , .

[UNIFIED-PLAN] Roach, A., Uberti, J., and M. Thomson, "A Unified Plan for Using SDP with Large Numbers of Media Flows", draft-roach-mmusic-unified-plan (work in progress), July 2013.

Authors' Addresses

Suhas Nandakumar
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Email: snandaku@cisco.com

Cullen Jennings
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Phone: +1 408 421-9990
Email: fluffy@cisco.com